Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Trabajo Fin de Grado

ADAPTACIÓN DE UN SISTEMA DE DETECCIÓN DE PERSONAS EN CÁMARAS OMNIDIRECCIONALES A DESCRIPTORES DEEP LEARNING

Autor: Nicolás García Crespo
Tutor: Pablo Carballeira López
Ponente: Jesús Bescós Cano

Junio 2019

ADAPTACIÓN DE UN SISTEMA DE DETECCIÓN DE PERSONAS EN CÁMARAS OMNIDIRECCIONALES A DESCRIPTORES DEEP LEARNING

Autor: Nicolás García Crespo
Tutor: Pablo Carballeira López
Ponente: Jesús Bescós Cano

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2019

Trabajo parcialmente financiado por el Gobierno de España bajo el proyecto
TEC2017-88169-R (MobiNetVideo)

Resumen

Resumen

En el campo de la visión por ordenador, las redes neuronales convolucionales (CNN) destacan en tareas de detección y clasificación de objetos debido a su capacidad de procesar información de casi cualquier tipo de datos que puedan representarse en un espacio de dos dimensiones, como las imágenes o incluso la música.

Este trabajo trata de mejorar la eficacia de un sistema de detección de personas en tiempo real con cámaras omnidireccionales y su capacidad de generalización, gracias a las ventajas que ofrecen las redes neuronales convolucionales pre-entrenadas frente a los descriptores de imagen clásicos. Debido a que entrenar un red es un proceso lento y costoso, usamos una red pre-entrenada porque ofrece buenos resultados y no existe la necesidad de ajustarla o re-entrenarla. Por tanto, el objetivo es incorporar en el sistema la posibilidad de extraer descriptores de estas redes.

El clasificador del sistema de detección de personas consiste en una malla de máquinas de soporte vectorial (SVM) repartidas por toda la imagen, con un área de actuación llamada fóvea. De esta manera, el clasificador es capaz de distinguir las diferentes distorsiones que se producen en una imagen omnidireccional en función de la localización espacial de un objeto o, en este caso, de una persona.

Los resultados finales se han obtenido extrayendo los descriptores desde alguna de las capas de las CNN pre-entrenadas más populares, como AlexNet, Densenet201, MobileNetV2 y VGG19. Estas redes han sido entrenadas con bases de datos de millones de imágenes y miles de clases, como ImageNet, que hacen posible su uso en este tipo de aplicaciones. Por último, se analizarán los resultados de cada una de las redes y capas y se medirá el rendimiento del sistema con estos descriptores.

Palabras Clave

Redes neuronales convolucionales pre-entrenadas, descriptores deep learning, extracción de características, SVM, cámaras omnidireccionales.

Abstract

In the field of computer vision, convolutional neural networks (CNNs) are highly effective in object detection and classification, due to their ability to process information from almost any type of data that can be represented in a two-dimensional space, such as images or even music.

This Proyect tries to improve the efficiency of a real-time people detection system with omnidirectional cameras and its generalization ability, thanks to the advantages of pre-trained convolutional neural networks as compared to classic image descriptors. Due to the fact that training a network is a slow and expensive process, we use a pre-trained network which offers satisfactory results and eliminates the requirement to adjust or re-train the network. Therefore, the objective is to include in the system the capability to extract descriptors from pretrained CNN.

The classifier of the people detection system consists of a support vector machines (SVM) grid distributed throughout the image, with an area of action called fovea. As a result, the classifier is able to distinguish the different types of distortion that take place in an omnidirectional image according to the spatial location of an object or, in such a case, a person.

The final results have been achieved by extracting the descriptors from some of the layers using some of the most popular CNN pre-trained neural networks, as well as AlexNet, Densenet201, MobileNetV2 and VGG19. These neural networks have been trained with databases with millions of images and thousands of classes, such as ImageNet, making it possible to use them in this type of applications. Finally, the results of each of the networks and layers will be analyzed and the performance of the system will be measured with these descriptors.

Key words

Pre-trained convolutional neural networks, deep learning descriptors, feature extraction, SVM, omnidirectional cameras.

Agradecimientos

Quiero dar las gracias en primer lugar a mi tutor Pablo, por su disposición a ayudarme, escucharme y orientarme. He aprendido mucho durante el desarrollo de este proyecto. Y a todos los profesores de la Escuela Politécnica Superior de la UAM que he tenido el placer de tener durante el grado.

Por otro lado, finalizar mis estudios no hubiese sido posible sin el apoyo incondicional de mi familia, a la que quiero agradecer todo lo que ha hecho por mí. El cariño y optimismo que me transmiten ha sido uno de mis pilares fundamentales.

Agradecer especialmente a Andrea sus ánimos, sus fuerzas y su compañía. Porque sin ella tampoco hubiera sido posible llegar hasta aquí. De nuevo, gracias.

A Pedro, que es un hermano para mí y siempre ha estado a mi lado, quién nos lo iba a decir.

Finalmente a mis amigos, porque han sido un apoyo esencial para mí durante estos años y a todos mis compañeros de universidad por las horas en la escuela, pero también fuera de ella.

Índice general

Lista de figuras

Lista de tablas

1 Introducción

A continuación se va a dar una primera introducción a las razones que motivaron el sistema de detección de personas, a los esquemas en los que se fundamenta y al marco actual en el que se encuentra la detección de personas mediante aprendizaje profundo para encontrar métodos e ideas de adaptación. Explicando así, la motivación de uso de los algoritmos de detección, extracción de características y modelos de entrenamiento que se han utilizado. Además de exponer la motivación del uso de redes neuronales profundas.

1.1. Motivación

Actualmente las cámaras y sistemas de videovigilancia tienen mucha demanda ya que son una parte esencial de la seguridad en hogares, instituciones y lugares públicos. Según el informe sobre el mercado mundial de videovigilancia [12], en 2018 el precio de mercado fue de 36,89 billones de dólares y se espera que crezca hasta 68,34 billones de dólares para el año 2023, con una tasa anual compuesta de crecimiento (CAGR) del 13.1 %. Este crecimiento se debe en gran parte al desarrollo de nuevas tecnologías y al perfeccionamiento de las ya existentes, como los algoritmos de detección basados en modelos de aprendizaje profundo, aplicados a las imágenes recogidas por cámaras de videovigilancia. Con estas técnicas se amplía enormemente el número de aplicaciones posibles de un sistema de cámaras de seguridad, ofreciendo más información en tiempo real sobre la realidad que capturan. Por ejemplo, la segmentación de objetos online [13], el reconocimiento de acciones humanas [14], la localización y detección de incendios [15],la biometría [16] o, como en este trabajo, la detección de personas.

En la década pasada, muchas de las investigaciones dedicadas al procesamiento de imágenes en el campo de la visión por ordenador para la detección de personas, utilizaban cámaras convencionales como dispositivos de grabación (a día de hoy también se siguen usando). Estas cámaras distorsionan poco las líneas y formas de objetos y personas. En consecuencia, con este tipo de imágenes se puede hacer uso de algoritmos de detección basados en ventanas deslizantes como HOG (histograma de gradientes orientados) [17], DPM (modelo de partes deformables) [18], HAAR [19], ACF (características agregadas de canal) [20] o LBP (patrones locales binarios) [21]. Se llegan a conseguir muy buenos resultados, no obstante, una de las principales desventajas de estos algoritmos es su coste computacional [22], ya que en estos esquemas, la ventana tiene que desplazarse a lo largo y ancho de cada imagen, calculando y comparando sus características

en esa ventana y en cada desplazamiento. Además, esto se realiza a diferentes factores de escala. Por otro lado, las lentes de las cámaras convencionales ofrecen un campo de visión (FOV) muy reducido en comparación con otro tipo de tecnologías. Estos dos problemas, entre otros, hicieron que se abrieran nuevas líneas de investigación.

El clasificador por excelencia fue la máquina de soporte vectorial (SVM), basado en técnicas de aprendizaje automático. Además existían otros como los árboles de decisión y las redes neuronales artificiales (ANNs) que se empezaron a utilizar en sistemas de detección de personas y objetos. Aunque en estos años, como se concluye en el trabajo de R. Benenson *et al.* [23], los resultados que se obtenían para la detección de personas con este tipo de redes no tenían una clara evidencia de ser buenas opciones ni ser superiores a otros algoritmos.

Las grandes bases de datos de las que disponemos, junto con el incremento del poder computacional del hardware de los ordenadores actuales, han permitido el éxito de esquemas basados en aprendizaje profundo para la detección con, típicamente redes neuronales convolucionales, en diversas tareas de la visión por ordenador. En paralelo, el estudio de diferentes tipos de cámaras tuvo como objetivo ampliar el campo de visión y debido a esto, se comenzaron a usar cámaras omnidireccionales. Estas cámaras tienen un campo de visión mayor que las cámaras convencionales (superior a 180^{o}), permitiendo cubrir áreas de un tamaño mayor. Por esta razón es muy interesante estudiar las posibilidades de uso de estas cámaras en los sistemas de videovigilancia. Como desventaja, este tipo de cámaras introducen fuertes distorsiones y varían en función de la posición del objeto en el campo de visión de la cámara [6]. Para solucionar el problema de la distorsión en estas cámaras, se utiliza un clasificador que consiste en una malla de SVMs repartidos en el espacio. Así cada clasificador se entrenará y detectará en base a la distorsión que haya en su zona del espacio. Por otro lado, el algoritmo que se usa para extraer los descriptores de las imágenes es HOG.

La motivación principal de este trabajo consiste en adaptar el sistema propuesto en [6] cambiando los algoritmos de extracción de características por otros basados en aprendizaje profundo, que puedan dar mayor robustez a los datos con el propósito de analizar cómo funciona el sistema con este nuevo esquema [24].

En los últimos años, muchos trabajos relacionados con la detección de personas han utilizado redes neuronales convolucionales (CNN), Region-Based CNN (R-CNN), Faster R-CNN o modelos híbridos de CNN+SVM [25] en los que se utiliza una red CNN pre-entrenada como herramienta de extracción de características para entrenar el clasificador. Estos han obtenido resultados por encima de muchos de los mejores modelos desarrollados hasta ahora [26]. Demostrando que este tipo de esquemas es, a diferencia de como se pensaba años atrás, claramente una buena línea de investigación para la detección de personas, [24].

Por estas razones, la adaptación del sistema de detección [6], se basará en algunas de estas técnicas, ya que el sistema utiliza un esquema basado en HOG+SVM, el cual puede ser modificado para adaptarse a estos nuevos modelos basados en aprendizaje profundo, que en un principio parecen dar buenos resultados.

1.2. Objetivos

Los dos objetivos principales de este trabajo son: (1) la adaptación a descriptores *deep learning* del esquema del sistema de detección de personas, proporniendo un nuevo esquema (2) Evaluación y análisis de su funcionamiento. Para llevar a cabo el objetivo principal se plantean las siguientes subtareas:

1. Análisis e investigación del funcionamiento y características principales de las redes neuronales profundas (DNNs), los tipos de redes neuronales convolucionales CNNs y los principales esquemas utilizados en el estado del arte.
2. Estudio y búsqueda de modelos de CNNs pre-entrenadas disponibles, así como las características y diferencias principales de cada una de ellas, con el objetivo de extraer vectores de características desde alguna de sus capas.
3. Evaluación final y comparación de resultados entre el sistema basado en HOG+SVM y el nuevo esquema propuesto CNN+SVM.

1.3. Documentación

La estructura del trabajo estará organizada de la siguiente manera. En el Capítulo 2, en la Sección 2.2 se describen y se profundiza en los algoritmos y esquemas utilizados en el estado del arte para cámaras de perspectiva, seguido de las principales diferencias entre estas y las cámaras omnidireccionales en la Sección 2.3. En el Capítulo 3, Sección 3.1, se describirá el sistema de detección propuesto en [6] y en la Sección 3.2 analizamos las principales arquitecturas de aprendizaje profundo propuestas para este trabajo y las adaptaciones realizadas en el sistema de detección de personas. El Capítulo 4, en las Secciones 4.1 y 4.2 haremos una introducción a los experimentos y a la base de datos. Después, en la Sección 4.4 veremos cómo se han obtenido los resultados globales y cuáles son. Finalmente, en el Capítulo 5 se hará una conclusión final sobre el trabajo y presentaremos algunos trabajos futuros que sigan la línea de investigación propuesta.

2

Estado del arte

2.1. Introducción

Durante los últimos años, la detección de personas ha sido uno de los problemas más estudiados en la visión por ordenador. Durante este tiempo el objetivo principal ha sido buscar detectores y algoritmos que proporcionaran más precisión. En el estado del arte existen muchos que pretenden cumplir con este objetivo. En este capítulo vamos a ver los detectores más utilizados y algunas de sus ventajas e inconvenientes. Debido al uso mayoritario de cámaras convencionales en hogares, sistemas de videovigilancia, coches,etc., estos esquemas están diseñados, en general, para trabajar con cámaras convencionales. Las cámaras omnidireccionales tienen características diferentes a estas y es por ello que el uso de algoritmos basados en ventanas deslizantes no es válido sin hacer modificaciones adaptadas a las características de las imágenes omnidireccionales, como la distorsión.

Haremos una revisión de las principales diferencias y finalmente veremos aquellos esquemas del estado del arte válidos para estas cámaras.

2.2. Algoritmos para la detección de personas basados en cámaras de perspectiva

Los algoritmos clásicos que destacan son aquellos basados en un esquema de ventanas deslizantes. Existen otros basados en regiones de interés o segmentación, aunque con peores resultados [27]. Durante años se han realizado muchos estudios para perfeccionar estos algoritmos. Vamos a centrarnos en analizar el funcionamiento y las ventajas y desventajas de los más importantes. El objetivo es entender los conceptos de estos algoritmos para poder relacionarlos con las posibles adaptaciones a cámaras omnidireccionales. Sin embargo, debido al gran desarrollo en el área de las redes neuronales, se han popularizado detectores basados en redes neuronales convoluciones (CNNs). La introducción del aprendizaje profundo hace que se consigan tamaños de descriptores menores, además de dotar al sistema de un aprendizaje robusto, de manera que secuencias nuevas no vistas por el sistema, consigan una buena precisión en la detección. Este es el motivo principal que justifica este trabajo.

Figura 2.1: Esquema de funcionamiento de ventanas deslizantes.

2.2.1. Esquema de ventanas deslizantes

Esta técnica era frecuentemente usada en multitud de algoritmos y aplicaciones muy diversas, no solo en el campo de la visión artificial. Actualmente, el método de mayor éxito se basa en utilizar bloques de aprendizaje profundo, aplicados a cualquier secuencia o conjunto de datos con el objetivo de controlar su flujo o extraer información. No obstante, los algoritmos que usan un esquema de ventanas deslizantes se siguen utilizando y es interesante analizar cómo funcionan para justificar su uso en diferentes tipos de cámaras.

El esquema general de ventanas deslizantes de la Figura 2.1 se basa en ir tomando diferentes regiones de una imagen con una ventana. El tamaño de la región será equivalente al tamaño de la ventana, definido por su ancho y alto en píxeles. La ventana se coloca en una esquina de la imagen y se va desplazando un número de píxeles definido por el algoritmo, con un solapamiento generalmente alto. La ventana recorre toda la imagen de izquierda a derecha y de arriba a abajo. De cada imagen contenida en las regiones se sacará un vector de características, es decir, habrá un descriptor por ventana.

En el estado del arte, el objetivo es evaluar si la imagen contiene o no a una persona. Dada la naturaleza de las imágenes convencionales, el tamaño de la persona puede ser diferente dependiendo de su posición en la escena. La solución consiste en redimensionar la imagen a varias escalas por un factor (pirámide multiescala), mientras el tamaño de la ventana no cambia. Así, alguien alejado en el fondo, podrá ser capturado por la ventana haciendo la imagen más grande.

Los descriptores obtenidos con las ventanas se pasan a la entrada de un clasificador. En la Subsección 2.2.3 se explica con más detalle la función de un clasificador y algunos de los más usados en el estado del arte. Estos pueden entrenarse con los descriptores o, en base a un pre-entrenamiento, determinar si cada uno de ellos contiene una persona o no. Una vez clasificados, la última etapa es de supresión de no máximos (NMS), que se encarga de descartar múltiples detecciones para una misma persona, quedándose solo con una. Finalmente, se puede saber si la imagen contiene algún objeto viendo si ha existido una detección positiva sobre alguna ventana.

En general y más aún si tenemos imágenes de alta resolución, el proceso es costoso computacionalmente, debido a que la ventana tiene recorrer toda la imagen y además para diferentes tamaños. Por eso es difícil conseguir implementar este algoritmo para aplicaciones en tiempo real, aunque existen técnicas de preprocesamiento para reducir el área de búsqueda ([28], [29]).

2.2.2. Extracción de características

Los humanos podemos identificar rápidamente a una persona en una imagen. Lo primero que hacemos es fijarnos de manera general en su forma y sus características. Los algoritmos para extracción de características hacen algo similar. Sin embargo, lo que ve un algoritmo es una matriz enorme de valores. Si nosotros tuviéramos que interpretar así la imagen sería muy

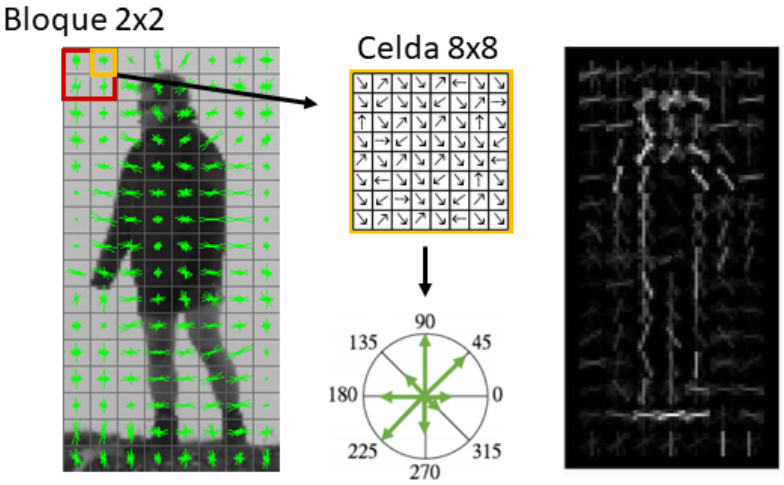

Figura 2.2: Descriptor HOG con bloques de tamaño 2x2, celdas de 8x8 con orientaciones del gradiente y resultado de la representación de los gradientes. Adaptado de [1].

costoso extraer información sobre su contenido. Para abordar el problema, los algoritmos se encargan de extraer un vector de valores que representen de la manera más generalista posible la información. Además, cuanto menor sea el tamaño del vector, más rápida será la detección en el clasificador. La forma en la que se obtienen estos valores es lo que varía en función del algoritmo.

En esta sección veremos algunos de los más importantes dedicados a la detección de personas.

Histograma de gradientes orientados (HOG)

Desde su introducción por Dalal y Triggs [17], HOG ha sido uno de los algoritmos más utilizados para obtener descriptores de una imagen en la detección de personas. Existen modificaciones del algoritmo para hacerlo de manera eficiente, incluso para aplicaciones en tiempo real [30]. Por sus buenos resultados la gran mayoría de los detectores en el estado del arte utilizan el algoritmo de HOG en alguna de sus formas.

Este algoritmo utiliza el esquema de ventanas deslizantes, donde la dirección del gradiente se calcula en base a la variación de intensidad de cada píxel respecto a su vecindario. Gracias a calcular solo la dirección en que varía la intensidad, una imagen muy oscura de una persona y otra muy clara de esa misma persona quedan representadas de la misma manera, i.e, es invariante a cambios de iluminación globales.

Para cada ventana se calcula el gradiente de todos sus píxeles. Después la región se divide en celdas de tamaño fijo, donde para cada celda se obtendrá un histograma. Los valores más altos del histograma se corresponden con los gradientes principales de cada dirección. Es común utilizar celdas de 8x8 píxeles para conseguir histogramas de 9 barras y orientaciones del gradiente entre 0^o y 180^o. Las celdas se agrupan en bloques Figura 2.2, preferiblemente solapados, donde se concatenan los histogramas que contienen las celdas de cada bloque. Dando como resultado final un de vector global de características que representa la imagen. El vector es el que se pasará a la entrada del clasificador para hacer la detección en función del modelo entrenado con otros vectores extraídos de la misma manera. Así, se consigue representar la información de la imagen con un conjunto de histogramas que además, permiten determinar la distribución espacial de los datos.

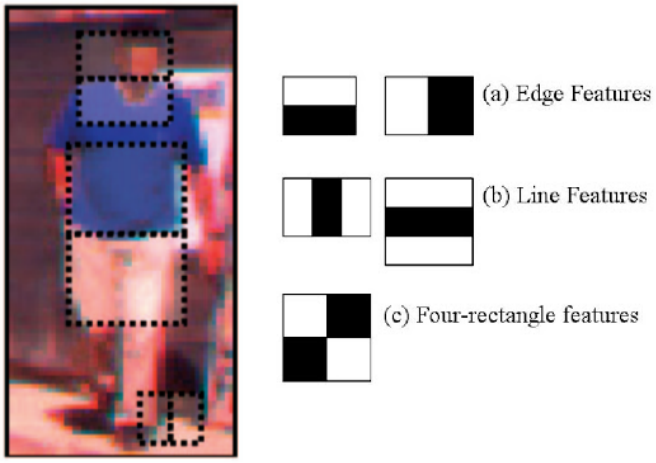

Figura 2.3: Banco de filtros básicos de HAAR y ejemplo de aplicación. Adaptado de [2].

Características HAAR

Las características HAAR se inspiran en los *wavelets* de Haar, de ahí su nombre. Su primer uso en detectores de personas fue en el trabajo realizado por Papageorgiou *et al.* en [19]. En el algoritmo de extracción de características HAAR existen unos filtros llamados filtros de Haar básicos. El filtro se aplica sobre la imagen y la va filtrando. Esto se hace en diferentes zonas de la imagen y a diferentes tamaños también, detectando cambios de intensidad en cualquier posición a diferentes escalas y orientaciones. Obtenemos así las respuestas del contorno de un persona en diferentes orientaciones. El resultado del filtrado se calcula como la diferencia entre la suma de intensidad de todos los píxeles situados en la zona positiva del filtro, menos la suma de intensidad de todos los píxeles situados en la zona negativa del filtro, normalizando el resultado. Así, la descripción global de toda la imagen es la concatenación de todas las características obtenidas.

Entendiendo cómo funciona el algoritmo, se puede ver que el número de características, y la dimensionalidad final del descriptor es muy elevada.

Otros

Existen otros algoritmos dignos de mención, pero no vamos a entrar en detalles sobre su funcionamiento. Esto son los patrones locales binarios (LBP), características de canal agregado (ACF) [20] y el modelo de partes deformables (DPM) [31]. Todos ellos basados también en el esquema de ventanas deslizantes de la Figura 2.1. LBP se utiliza en sistemas de detección de personas en combinación con otros algoritmos, como en [32], [33] o [34]. Para más información sobre las aplicaciones de ACF ver [35].

2.2.3. Clasificadores basados en aprendizaje automático

Un clasificador es el que se encarga de analizar los descriptores. Por ejemplo, usando los que se han extraído con el método de ventanas deslizantes de la Sección 2.2.1. Su función es determinar en base a los datos que contienen los descriptores, si existe o no un objeto en las imágenes. Es necesaria una fase de entrenamiento donde el clasificador aprende lo que es un objeto o una persona. El clasificador más popular con diferencia en la década pasada fue la máquina de soporte vectorial (SVM), que a día de hoy sigue siendo uno de los más importantes en el estado del arte. Todos los detectores que vamos a ver a continuación son de aprendizaje

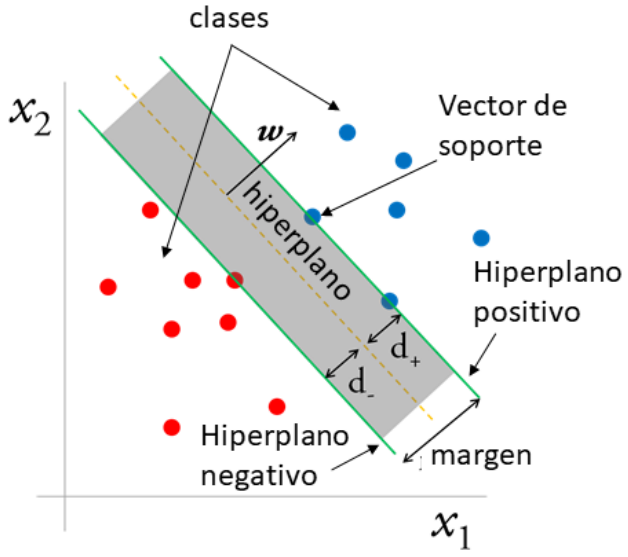

Figura 2.4: Esquema de una máquina de soporte vectorial SVM. Adaptado de [3].

supervisado, definiendo la localización de las personas en una imagen con etiquetas definidas por el usuario.

Máquina de soporte vectorial (SVM)

El SVM es un algoritmo de aprendizaje automático desarrollado por V. Vapnik [36]. Se utiliza en problemas de clasificación y regresión.

En la fase de entrenamiento de este clasificador se utilizan como entrada los descriptores obtenidos con algún algoritmo de extracción de características, junto con las etiquetas de clase de cada descriptor, por ejemplo, una clase para indicar la presencia de una persona y otra si no la hay. Existen diferentes kernels que representan los datos de entrada en un espacio multidimensional. El kernel lineal es con el que mejores resultados se obtienen, mientras que otros como los polinomiales o sigmoides no son tan eficientes. Lo que se trata de calcular es un hiperplano que separe en el espacio multidimensional los datos de entrada Figura 2.4, es decir, las dos clases, minimizando una función de coste basada en una medida de distancia entre el hiperplano y los datos. El número de clases puede ser mayor y varía en función de la finalidad del clasificador, pero el caso de la detección de personas solo hay dos clases posibles. De esta manera, en la fase de detección, el descriptor de la imagen que se quiere clasificar quedaría representado de la misma manera en el espacio y dependiendo de en qué lado del hiperplano cayera, se clasificaría como perteneciente a una clase u otra.

El clasificador puede ser ajustado por el usuario mediante dos parámetros. El primero es el parámetro de regularización, comúnmente denotado por C. Este parámetro define cuánto se ajusta el hiperplano a las dos clases. Dependiendo de como se defina se podría sobre-entrenar el clasificador. El segundo es el margen de separación, que puede ser positivo o negativo, Figura 2.4. Se busca que la distancia al hiperplano, de las muestras de cada clase más cercanas a este, sea lo más parecida posible. La distancia a la que queda el hiperplano de estas muestras se conoce como margen positivo y margen negativo, según la clase de la muestra. Esta distancia se puede cambiar, moviendo todo el plano hacia una clase u otra, dando así mayor prioridad a una clase. Si una muestra cayera próxima al hiperplano, esta podría provocar un falso positivo o, ajustando este parámetro, un verdadero positivo.

Este detector puede formar parte de la etapa de clasificación de un sistema, siendo muy popular utilizarlo en combinación con algoritmos como HOG u otros que hemos visto en la Sección 2.2. En el sistema de detección propuesto en este trabajo, es precisamente este clasificador el que se utiliza, pero con algunas particularidades que veremos más adelante.

2.2.4. Arquitecturas *deep learning*

Para entender los esquemas que vamos a ver a continuación es necesario hacer una pequeña introducción al concepto de neurona y red neuronal artificial (ANN). Las ANN están inspiradas en la forma en que las neuronas de la corteza visual cerebral se conectan entre sí [37], de ahí su nombre. Se puede entender como un modelo dividido en diferentes capas compuestas por varias neuronas conectadas entre sí, que se van modificando y se van pasando la información de unas a otras.

La neurona es la unidad mínima de una red. Está compuesta por una serie de entradas, unos pesos, el bias y una función de activación (ReLU, Sigmoide, Lineal, Tangencial o Softmax). Esta función de activación debe aplicar necesariamente una transformación no lineal a la entrada para poder hacer a la red capaz de aprender. El conjunto de capas de la red se divide en los siguientes tipos.

- **Capas de entrada**, aquellas que reciben la información de entrada a la red.
- **Capas ocultas**, formadas por conjuntos de neuronas que realizan diferentes cálculos en función de sus parámetros.
- **Capas de salida**, que son las que contienen el resultado final de las operaciones matemáticas realizadas por las capas anteriores.

El número de capas ocultas y su capacidad, es decir, el número de neuronas que hay en una capa, se puede modificar según el diseño. Si una red tiene muchas capas ocultas recibe el nombre de red neuronal profunda, de ahí el título que engloba a estas redes, redes de aprendizaje profundo o *deep learning*. Todas las capas de una ANN están formadas por capas totalmente conectadas. Son aquellas donde las neuronas de una capa se conectan con todas las neuronas de la siguiente capa. El peso de una conexión se multiplica por la salida de una neurona, cuyo resultado será la entrada de la neurona de la capa siguiente. Suele ser una de las capas finales de la red y es la que se encarga de la detección o clasificación. A continuación, veremos algunas arquitecturas de redes basadas en aprendizaje profundo, cuyas capas no son todas totalmente conectadas. Es precisamente esto lo que hace interesante el uso de estas redes en el problema que nos ocupa.

- **Red neuronal convolucional (CNN)**.

  Las redes neuronales convolucionales están basadas en el Neocognitron, introducido por Fukushima en [38]. De alguna manera, estas redes están. Las redes neuronales convoluciones son básicamente una modificación de las redes neuronales artificiales para trabajar con imágenes en dos dimensiones. Por esta razón son tan utilizadas en la visión por ordenador para la detección de personas, patrones y casi cualquier tipo información que pueda ser representada en dos dimensiones, como el audio [39]. Son especialmente buenas en caracterizar y capturar datos locales próximos espacialmente y relacionados entre sí.

  La arquitectura de estas redes se puede ver en la Figura 2.5 y se divide en diferentes capas. Las capas convolucionales tienen un grupo de kernels que se convolucionan con la imagen. La operación de convolución en estas capas, con un conjunto de filtros K de dimensión

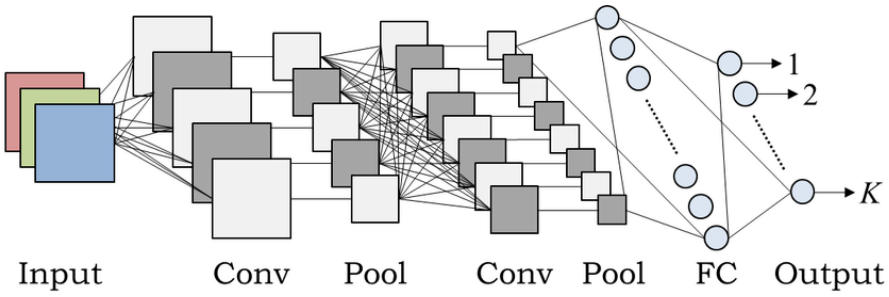

Figura 2.5: Arquitectura de una red neuronal convolucional CNN. Extraído de [4].

cuadrada *(2d + 1) × (2d + 1)*, donde d es el tamaño del vecindario sobre el que se va a aplicar la fórmula, una señal de entrada X y el filtro H, se define como:

$$Y_k[i,j] = \sum_{n=i-d}^{i+d} \sum_{m=j-d}^{j+d} X[n,m]H_k[n-i,m-j], k = 1,...,K \tag{2.1}$$

Este proceso se conoce como filtrado. La salida de una capa convolucional será un número de imágenes igual al número de filtrados que se hayan realizado. El tamaño de las imágenes de salida dependerá del tamaño de los filtros. Dado que a lo largo de la red existen varias capas de este tipo es necesario ir reduciendo la resolución de las imágenes. Para esta labor están las capas de *max pooling*. Lo que hacen es ir desplazando una ventana sobre cada imagen de salida de una capa convolucional y se quedan con el máximo valor de entre todos los que se encuentran dentro de la ventana. Así se consigue reducir el tamaño de las imágenes, pero el número de canales sigue siendo el mismo.

Los valores que se obtienen como resultado de estas capas suelen tener valores negativos. Las capas *ReLU* se encargan de poner todos los valores negativos a cero. Finalmente se encuentran las capas *fully-connected*, que ya hemos definido. Todos los valores que llegan a esta capa influyen en la decisión final.

Los hiperparámetros de estas redes son el número total de capas, el número y el tamaño de los kernels de las capas convolucionales, el tamaño y el paso de la ventana de las capas max pooling y el número de neuronas de las capas totalmente conectadas.

En los siguientes puntos veremos redes que funcionan como detectores basados en modificaciones de las CNN, muy utilizadas en el estado del arte.

- **RCNN**: Las RCNN (Regiones con características CNN) son redes populares en la detección de objetos. El método de funcionamiento se puede entender como si la red, para realizar las detecciones, generara un cuadro de separación en la imagen e identificara si lo que este contiene es un objeto o no. Con el fin de poder localizar dicho objeto en la imagen. El esquema general puede verse en la Figura 2.6, imagen izquierda.

  Una de las principales características de este tipo de redes es que su salida no sigue una distribución lineal, porque se quiere detectar cualquier tipo de objeto en la imagen y el número de estos puede variar. Por eso no es posible utilizar una red CNN convencional para este tipo de problemas. Lo que sí es posible sería dividir la imagen de entrada en regiones de interés y utilizar una CNN para clasificar el objeto de cada región. El problema es que hacer esto tiene un coste computacional muy alto. Como solución se proponen esquemas que modifican esta arquitectura, más rápidos y eficientes, como las Fast RCNN o Faster RCNN.

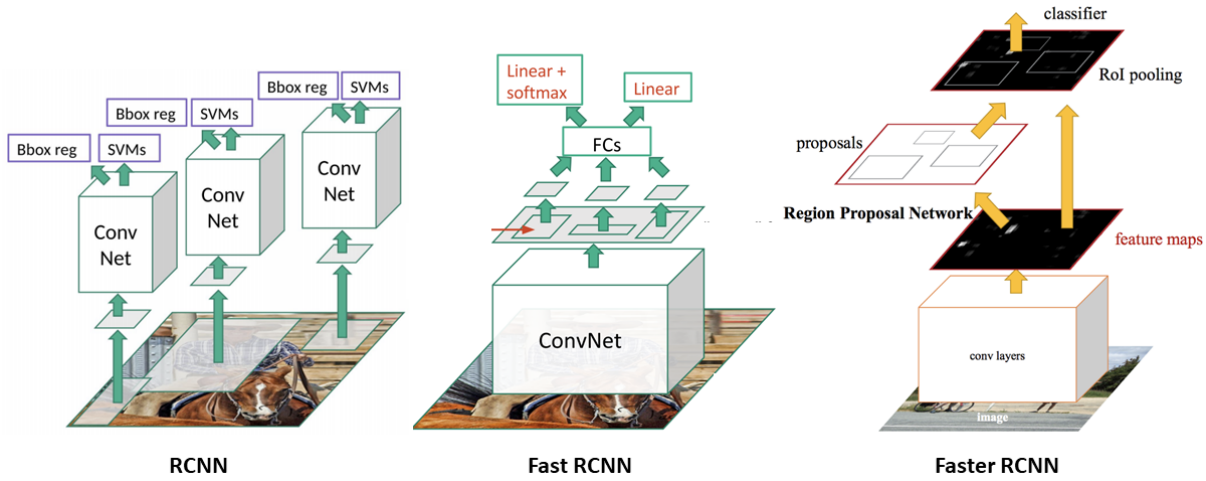

Figura 2.6: Arquitecturas de redes neuronales convolucionales basadas en regiones. Extraído de [5].

Ross Girshick *et al.* en [40] propusieron un método donde se definía la arquitectura de la RCNN. El problema de tener que clasificar un número enorme de regiones se soluciona dividiendo la imagen en solo 2000 regiones, llamadas regiones propuestas. Esta división se realiza con algoritmos de búsqueda selectiva para reconocimiento de objetos, la información detallada sobre este algoritmo se puede ver en [41]. Las regiones propuestas serán la entrada de una CNN, que actuará como método de extracción de características. Los descriptores se pasan a un SVM para clasificar el objeto que se encuentra dentro de cada región propuesta. Adicionalmente se calculan otros valores para mejorar la precisión de las cajas delimitadoras.

Las RCNN no son todo lo rápidas que podrían ser, ya que a fin de cuentas, tienen que procesar 2000 regiones. El tiempo de procesamiento que se obtuvo para una sola imagen en [40] es de 47 segundos. Además, el algoritmo de búsqueda selectiva no se puede adaptar a unos datos de entrenamiento, lo que puede provocar que se propongan malas regiones.

- **Fast RCNN**: Estas redes son una modificación de las RCNN cuyo esquema es propuesto por el mismo autor en [42] para mejorar el tiempo de procesamiento de las imágenes. En vez de utilizar la imágenes de entrada en un algoritmo de búsqueda selectiva que determine las regiones propuestas, estas imágenes serán la entrada de una CNN, que a su salida creará un mapa de características convolucional, Figura 2.6. Es con este mapa con el que se identificarán las regiones propuestas usando el algoritmo de búsqueda. Las regiones serán redimensionadas para que puedan ser la entrada a una capa totalmente conectada, cuya salida formará los vectores de características de interés (*RoI features*). Finalmente se clasifican con una capa softmax y se extraen los valores característicos de los boundary boxes. Los cálculos sobre las imágenes de entrada en la CNN consisten en una sola convolución. En vez de proponer regiones se extrae un completo mapa de características. El tiempo de procesamiento sin tener en cuenta las regiones propuestas, en comparación con las RCNN, es aproximadamente 140 veces más rápido, unos 0.3 segundos. Teniendo las regiones propuestas en cuenta se tiene 2.3 segundos de procesamiento [42].

- **Faster RCNN**: El esquema propuesto por Shaoqing Ren *et al.* en [43] no utiliza un algoritmo de búsqueda selectiva para encontrar las regiones propuestas, ya que este tipo de algoritmos ralentizan el rendimiento de la red. Al igual que en las Fast RCNN, se utiliza primero una CNN a la entrada para conseguir el mapa de características, Figura 2.6. En este caso, una red separada del resto de la arquitectura se encarga de obtener las regio-

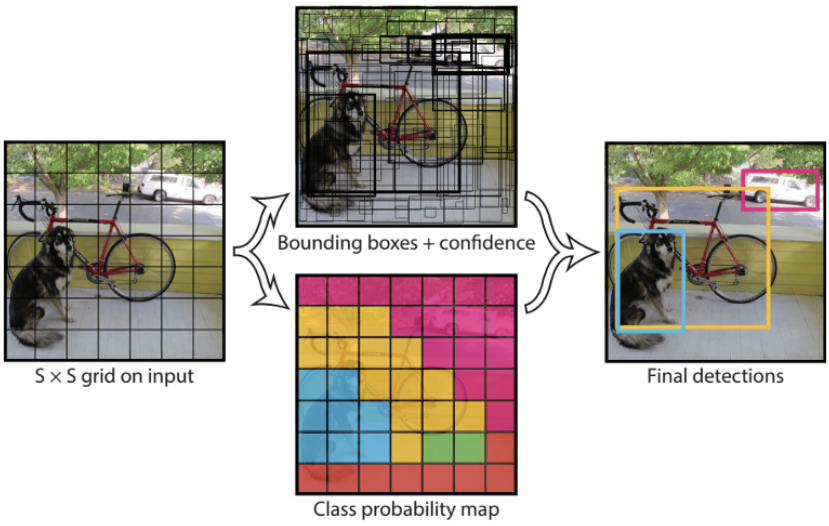

Figura 2.7: Detección de objetos con YOLO. Extraído de [5].

nes propuestas, permitiendo además a las Faster RCNN aprender sobre las regiones. Las regiones obtenidas se redimensionan para que una RoI pooling layer clasifique la imagen. Finalmente, estas redes son capaces de procesar las imágenes en tan solo 0.2 segundos.

- **YOLO**: A diferencia de las redes que hemos visto, YOLO (You Only Look Once) no utiliza regiones para localizar los objetos en la imagen. El funcionamiento consiste en dividir la imagen en bloques de NxN píxeles. Para cada bloque se definen M boundary boxes. Las imágenes contenidas en estas cajas se pasan por una CNN que devuelve valores de offset y la probabilidad de clase de cada caja. Finalmente se establece un umbral por encima del cual se detecta y localiza un objeto. Esta arquitectura está formada por una sola CNN, que determina las boundary boxes y la probabilidad de clase del objeto que se encuentra dentro de cada caja.

  YOLO es capaz de procesar una imagen cada 0.02 segundos, [44]. Sin embargo, su principal problema aparece cuando tiene que detectar objetos pequeños, debido a que no identifica en profundidad las características espaciales de la imagen. Esta arquitectura está muy presente en el estado del arte. En varias aplicaciones consigue destacar, como en sistemas de detección en tiempo real [45].

2.3. Cámaras convencionales y omnidireccionales

Todas las arquitecturas que hemos visto en la Sección 2.2.4 tienen una capa final dedicada a la clasificación y detección del objeto y son muy utilizadas en el estado del arte. Pero debido a la naturaleza de los datos de entrenamiento y que asumen una posición invariable con respecto a la posición de la imagen, no es posible su implementación para el desarrollo de este trabajo. Las cámaras omnidireccionales presentan características diferentes a las imágenes con las que se entrenan estas redes. Por eso hay sistemas propuestos con algunas modificaciones sobre los esquemas que se han estudiado en este capítulo para abordar el problema. En esta sección se estudiarán las principales características de las imágenes convencionales y omnidireccionales para después, en el siguiente capítulo introducir el esquema del sistema propuesto.

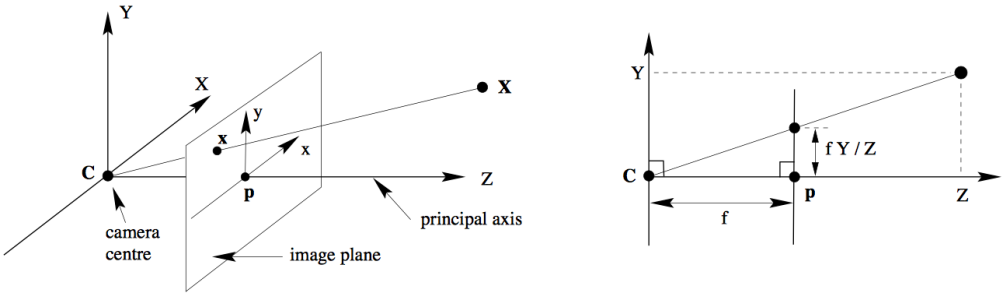

Figura 2.8: Modelo de cámara pinhole y proyección de perspectiva. Extraído de [6].

2.3.1. Perspectiva, proyección y tipos de cámaras

En primer lugar, para introducir el funcionamiento y las características principales de una cámara, definiremos el modelo en el que se basan las cámaras de perspectiva o convencionales. Estas cámaras recogen la luz del entorno y son capaces de recoger una imagen tal y como el sistema visual humano observa la realidad, aunque con algunas pequeñas diferencias que veremos a continuación. El modelo que describe estas cámaras es el modelo de cámara puntual *pin hole* [46]. Este modelo se puede entender como una caja con un orificio por el que los rayos de luz pasan para proyectar la imagen invertida en el lado opuesto de la caja. En este caso, todos los haces de luz pasan a través de un punto 3D en común, que define el centro de proyección (COP) de la cámara o punto de vista efectivo [6]. En este caso, la imagen se proyecta de forma inversa porque las cámaras de perspectiva solo tienen un centro de proyección. La naturaleza de este modelo puede describirse fácilmente como una proyección tridimensional, donde la linealidad se conserva al proyectar la imagen, Figura 2.8. Sin embargo, se introducen algunas distorsiones consecuencia de los ángulos y las distancias, que no se mantienen.

Esta es una primera aproximación para describir imágenes de perspectiva, pero las cámaras realmente utilizan un modelo más complejo conocido como modelo de lente delgada. La dirección de los rayos que atraviesan la lente se rigen por la Ley de Snell, basada en los índices de refracción del aire y la lente.

En el modelo de lentes se define la distancia focal como la longitud de separación entre el centro óptico de una lente y el foco. Este último es el lugar donde un punto 3D se proyecta y converge en un punto 2D llamado P' [47]. La forma curvada de la lente puede introducir algunas distorsiones cromáticas, esféricas o geométricas como la distorsión de barril o la distorsión pincushion. Se suelen utilizar varias lentes para reducir la distorsión, pero aún así estas no son especialmente graves.

Si el problema de las cámaras convencionales no es la distorsión, entonces habrá una razón para estudiar otro tipos de cámaras. El campo de visión (FOV) es el factor más limitante de este tipo de cámaras. Se define como el ángulo que es capaz de abarcar la cámara para capturar el entorno. El sistema visual humano tiene un campo de visón aproximado de 150 grados, mientras que las cámaras convencionales tienen tan solo unos 50 grados.

Los tipos de cámara pueden clasificarse en función de su campo de visión. Las cámaras que son capaces de cubrir un gran ángulo en una dirección son panorámicas, mientras que aquellas que son capaces de cubrir todo el espacio en todas las direcciones, con un campo de visión de 360º, son omnidireccionales. Sin embargo, muchas veces se suele referir a las cámaras omnidireccionales como panorámicas, ya que ambas tienen un campo de visión grande.

La manera de conseguir campos de visión mayores es variada. Desde sistemas multicámara formados por la unión de varias cámaras de perspectiva, donde cada cámara se encarga de capturar un área del espacio. Pasando por modificaciones de lente como las ojo de pez. Hasta

Figura 2.9: A la izquierda una imagen obtenida con una cámara de espejo curvo catadióptrica. A la derecha una imagen obtenida con una cámara ojo de pez. Extraído de [7].

sistemas complejos donde el grupo óptico está compuesto por un conjunto de lentes y un espejo curvado que hace una función parecida a la deformación de las lentes de ojo de pez, llamadas cámaras catadióptricas. Sobre estos y más sistemas [46].

Las cámaras omnidireccionales como tal, con un FOV de 360º, son complejas y costosas. Por eso suelen utilizarse cámaras panorámicas con grandes FOV. Las más utilizadas son las cámaras de ojo de pez y las catadióptricas. Cada una tiene sus particularidades: en las catadióptricas el espejo curvo provoca un punto ciego en el centro de la imagen, mientras que las cámaras de ojo de pez introducen distorsiones más fuertes en las zonas exteriores de la imágenes en comparación con las catadióptricas, representado en la Figura 2.9. Al fin y al cabo, ambas capturan imágenes similares y a partir de ahora, por tener campos de visón grandes, nos referiremos a ellas como cámaras omnidireccionales.

2.3.2. Imágenes omnidireccionales

Las imágenes omnidireccionales son capaces de cubrir áreas mucho mayores en comparación con las convencionales. Esta es la principal motivación por la que existen muchos algoritmos dedicados a este tipo de imágenes en el estado del arte, como en [48] o [49]. Un sistema así formado por cámaras convencionales sería mucho más caro y costoso, ya que se necesitarían más cámaras para abarcar el área de una misma sala. Si en un sistema de seguridad tenemos más cámaras el proceso de sincronización y calibración se hace mucho más complejo. Si con las cámaras omnidireccionales conseguimos reducir el número de cámaras, también se reduce el número de imágenes que se tienen que manejar y procesar. Simplemente grandes cantidades de datos son más complejos de gestionar.

Sin embargo, las imágenes omnidireccionales imponen también algunos problemas. El principal inconveniente son las graves distorsiones que provocan en la forma de los objetos de la imagen. Estás distorsiones además, no son uniformes, es decir, que el tipo de distorsión varía en función en la localización espacial del objeto en el entorno. La relación de aspecto se pierde totalmente. Por eso, la resolución espacial en la imagen varía en función de la posición del objeto, es decir, de su distorsión. En algunas cámaras la resolución espacial en la periferia de la imagen será peor, mejorando en dirección al centro de la imagen, como en las cámaras de ojo de pez y en otras, ocurrirá lo contrario, perdiendo espacialidad en el centro de la imagen. Se tiene que tener también en cuenta que las imágenes de un sistema de videovigilancia suelen ser tomadas en interiores y desde una posición elevada, como por ejemplo en alguna esquina superior de una sala, como se puede ver la Figura 2.10.

Figura 2.10: Imagen de la base de datos PIROPO capturada con la cámara omnidireccional OMNI 1A. Extraído de [8].

.

Podemos deducir lo siguiente. La apariencia, en cuestión de la forma, de una persona en una imagen de perspectiva es en general invariante respecto a su posición en la imagen, solo puede cambiar su tamaño. Por eso los clasificadores dedicados a este tipo de cámaras no tienen que tener en cuenta su posición. Sin embargo, en imágenes omnidireccionales, la apariencia de la persona es totalmente diferente en el centro o en los bordes de la imagen. Los clasificadores en este caso deben considerar las diferentes formas que adoptan las personas en las diferentes partes de la imagen. Podríamos resumirlo en que si tenemos una sola posible apariencia, usaremos un clasificador y en cambio, si tenemos diferentes apariencias, necesitaremos varios clasificadores. Para el sistema de detección que nos ocupa es necesario también que las cámaras sean estáticas.

Conociendo mejor las principales características y dificultades que plantean las cámaras omnidireccionales podemos empezar a proponer soluciones. En el siguiente capítulo veremos el esquema de funcionamiento del sistema de detección de personas y cómo se consigue hacer que funcione con imágenes omnidireccionales.

3

Descripción del sistema de detección de personas

En este capítulo se verá con más detalle el esquema de funcionamiento del sistema de detección de personas en cámaras omnidireccionales [6] y sus etapas de entrenamiento y detección.

3.1. Esquema del sistema

El objetivo principal del sistema es la implementación de un esquema capaz de lidiar con los problemas inherentes a las imágenes omnidireccionales y trabajar en tiempo real para la detección de personas y objetos. En esta sección lo veremos un poco más en detalle y analizaremos cómo se estructuran los clasificadores para hacer las detecciones atendiendo a las características de las imágenes omnidireccionales.

Las distorsiones son diferentes para imágenes de perspectiva y omnidireccionales, como ya analizamos en la Sección 2.3.2. Para poder hacer frente a este problema, el esquema del sistema se basa en una malla de clasificadores SVM distribuidos por toda la imagen. Dependiendo del lugar donde se encuentren, cada uno de ellos atenderá únicamente a una zona diferente. De esta manera, cada SVM estará especializado en clasificar objetos en función de su localización con una determinada distorsión, igual que pasa en las imágenes omnidireccionales. Con un descriptor global para cada imagen, se obtendrá una predicción para cada clasificador, Figura 3.1. Finalmente, podrán recogerse todas las salidas y en función de qué clasificadores se hayan activado, se detectará la posición del objeto o la persona en la imagen.

En el trabajo realizado por L. García en [6], se implementaron dos sistemas de detección diferentes. Estos comparten las etapas de entrenamiento y detección. Lo que varía es la forma en que se extraen los descriptores. En el trabajo se refieren a uno como detector de personas en movimiento (MovPD), focalizado en la detección de objetos en movimiento y al otro, como detector de personas (PD), dedicado a la detección de objetos estáticos y en movimiento. Ambos aparecen representados con cuadrados rojos en la Figura 3.1. El sistema PD extrae los descriptores con HOG, mientras que el sistema MovPD lo hace primero con un preprocesamiento de las imágenes que hace una sustracción del fondo y después la extracción de descriptores con características HAAR, atendiendo a los resultados obtenidos en el paso anterior. En los dos casos hay un descriptor por imagen y es común para todos los clasificadores.

Las características de este sistema hace que se diferencie claramente de los detectores que

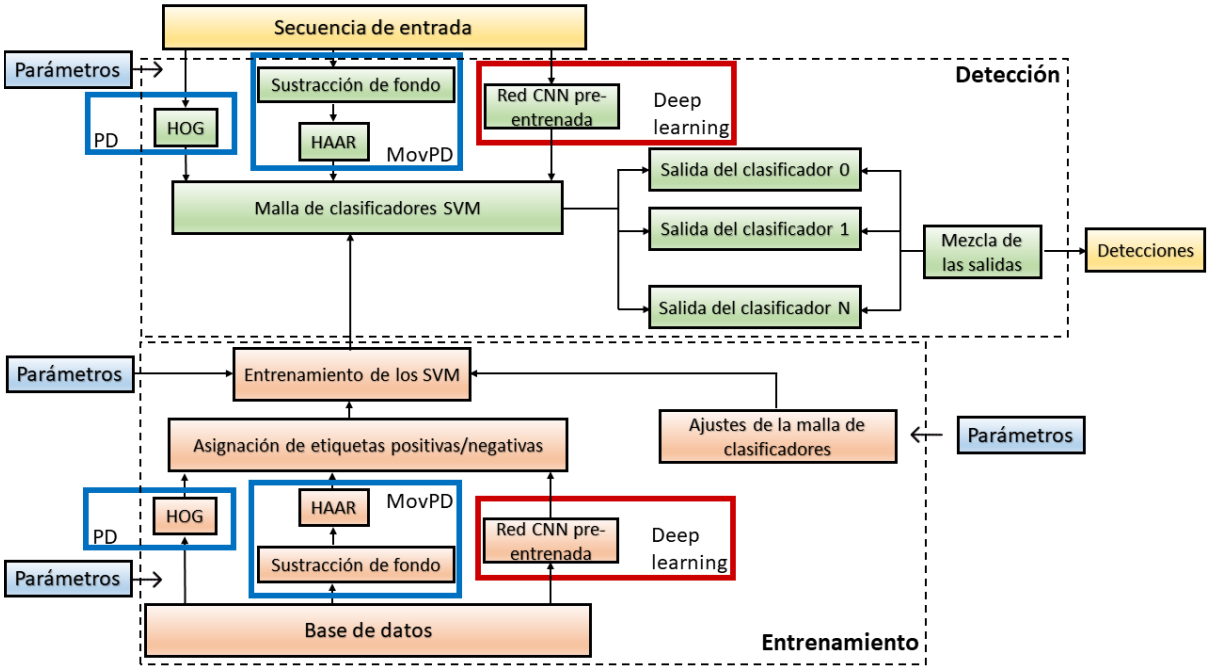

Figura 3.1: Esquema del sistema de detección de personas. Extracción de características HOG y HAAR recuadrado en azul y adaptación del bloque para descriptores *deep learning* recuadrado en rojo.

existen en el estado del arte. Su esquema hace capaz al detector de trabajar en tiempo real, ya que el coste computacional en el proceso de extracción de características es menor y además, a pesar de haber más clasificadores, el número de detecciones que se realizan es igual o incluso menor a las que se realizan en detectores basados en ventanas deslizantes. No obstante, el tamaño de los descriptores que es obtiene es grande y es necesario un ajuste de parámetros para obtener un buen resultado de precisión sin comprometer el coste computacional. En las siguientes secciónes veremos más a fondo las etapas de extracción de características, entrenamiento y detección para entender como se hace posible.

3.1.1. Extracción de características en el sistema

Como hemos visto en la Sección 2.2.1, el esquema basado en ventanas deslizantes no es la mejor manera de implementar un sistema capaz de trabajar en tiempo real. Por eso, no se utiliza este método para el proceso de extracción de características. Normalmente, un algoritmo basado en ventanas deslizantes obtiene un descriptor para cada ventana. Luego, un clasificador realiza la detección para cada uno de los vectores, decidiendo si en ese descriptor que representa la ventana existe una persona o no. En este caso no se realiza un escaneo de la imagen con ventanas, sino que en primer lugar la imagen se divide en bloques y se obtiene un descriptor global de toda la imagen, concatenando los descriptores obtenidos en cada bloque. El tamaño de los bloques es variable y existe la posibilidad de hacer una pirámide multiescala. En total habrá tantos descriptores como número de frames contenga el vídeo.

La técnica de sustracción de fondo para el sistema MovPD se basa en el modelo de mezcla gaussiano de Zivkovic en [50]. El objetivo de las sustracción es diferenciar los píxeles que pertenecen a un objeto en movimiento de los que pertenecen al fondo. Lo que se obtiene son imágenes binarias, donde los píxeles pertenecen al fondo o al objeto en movimiento. Esta técnica presenta algunas dificultades, por ejemplo, si una persona en algún momento del vídeo se detuviera, se consideraría que forma parte del fondo de la imagen. Además, para el algoritmo es complicado

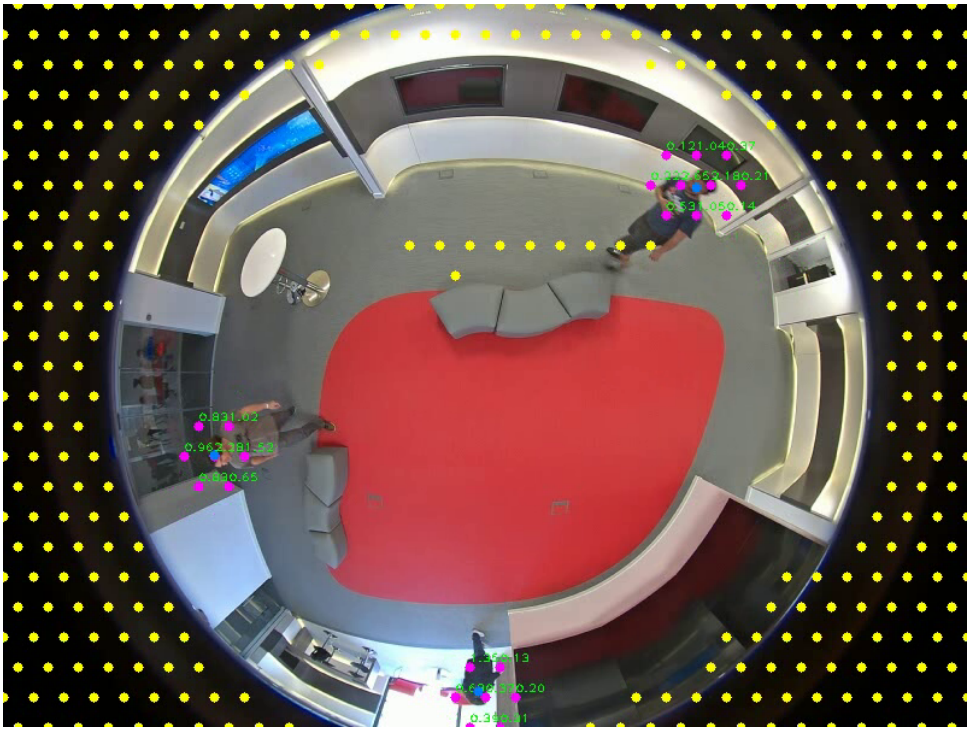

Figura 3.2: Detección realiza en el sistema de detección de personas con una malla hexagonal, 825 clasificadores, *Np*=7 y descriptores *deep learning*. Los puntos amarillos son los clasificadores no entrenados, los morados son los clasificadores activados, el punto azul es la detección que resulta de la interpolación de los clasificadores activados y los valores en verde representan el valor devuelto por cada SVM activado.

diferenciar los píxeles correspondientes a un objeto en movimiento de los píxeles de su sombra o reflejos. Es más fácil que el clasificador se equivoque a la hora de realizar la detección, por eso este sistema obtuvo en general peores resultados que el sistema PD para la detección de personas. En cambio, la extracción de características HAAR es simple, rápida y adecuada para utilizar sobre imágenes como las extraídas en la sustracción del fondo.

El sistema PD utiliza vectores de características HOG. Como hemos visto en el estado del arte, sección 2.2.2, este tipo de descriptores son muy populares y aportan robustez a los datos. Con estos descriptores el sistema es capaz de detectar objetos inmóviles, como personas paradas de pie o sentadas. Para el diseño del sistema se define el tamaño de las celdas para HOG en 8x8 píxeles, un tamaño de bloque de 16x16 píxeles y un desplazamiento de 8x8 píxeles, para de esta manera obtener histogramas de nueve orientaciones.

Dada la adaptación que se propone en este trabajo a descriptores deep learning, es más interesante desde el punto de vista formativo el sistema PD, ya que la finalidad del sistema es la misma.

3.1.2. Etapas de detección y entrenamiento

Cada uno de los clasificadores de la malla presenta un área de actuación, llamada fóvea (Figura 3.3). En este área es donde se produce la detección de las personas. La representación de estas áreas se describe con un punto en la imagen, formando una malla de puntos. La distribución de estos puntos se puede ajustar a diferentes patrones: rectangular, hexagonal y polar. También se puede definir el número de clasificadores. En el trabajo estudiado [6], se concluye una configuración del patrón y un número de clasificadores que se adaptan mejor al entorno y

Figura 3.3: Patrón hexagonal de la malla de SVM en la que cada punto verde representa la fóvea o área de actuación de cada clasificador. Extraído de [6].

ofrecen mejores resultados. Estos son el patrón hexagonal y 825 clasificadores.

En la malla se forman grupos de *Np* clasificadores, llamados vecindarios. Para el caso particular del patrón hexagonal $Np = 7$, como en la figura 3.2. Para la detección final se tendrán en cuenta todos los votos del vecindario que ocupe la región donde se encuentre la persona. El proceso es el siguiente:

1. Se aplica un umbral a un valor global que es calculado sumando las confidencias de cada clasificador del vecindario.
2. Se aplica supresión de máximos locales para eliminar múltiples detecciones de una misma persona
3. El punto que representa la posición donde se ha detectado la persona es el resultado de una interpolación del vecindario. Esta interpolación se calcula en base a los pesos de predicción positivos α_i y las coordenadas x_i, y_i de cada clasificador Ci:

$$Coord.\ del\ punto\ detectado\ [x,y] = \{\sum_{i=0}^{N_n} \alpha_i \cdot x_i, \sum_{i=0}^{N_n} \alpha_i \cdot y_i\},\ i = 0, ..., n \tag{3.1}$$

En la etapa de entrenamiento la manera en la que se entrenan los SVM es con el aprendizaje supervisado basado en puntos, es decir, con un *ground truth* que indica en qué píxel de la imagen se encuentra la cabeza de la persona. A diferencia de los detectores basados en ventanas deslizantes, que en su entrenamiento se etiquetan la ventanas. Además, siguen compartiendo un vector global como descriptor de la imagen. Sin embargo la etiqueta, positiva o negativa, será diferente para cada clasificador. Dijimos que los (*ground truths*) indicaban en qué píxel de la imagen se encuentra la cabeza de la persona. Lo que se hace en el entrenamiento es marcar la imagen con una etiqueta, positiva para aquel vecindario de *Np* clasificadores más cercanos al punto que indica la posición de la persona. Mientras que para el resto de clasificadores la etiqueta será negativa. Para reducir el coste computacional, los SVM que no han adquirido ninguna etiqueta positiva durante el proceso no se entrenarán, estos son los puntos amarillos en la Figura 3.2. La razón principal de hacer esto es que en las cámaras omnidireccionales existen zonas de la

imagen que no tienen ningún tipo de actividad y además la estructura del escenario hace que no sea posible que haya un objeto en determinados puntos. Estas zonas negras son producto de la manera en la que se tienen que representar las imágenes capturadas por cámaras con un campo de visión grande. Se pueden evitar así algunos falsos positivos durante la detección. Lo más interesante es fijarse en que gracias a este método de etiquetas, cada clasificador, dependiendo de la posición en la que se encuentren, aprenderá qué descriptores definen la forma de una persona y qué descriptores definen el fondo en ese mismo punto.

3.2. Adaptación del sistema a descriptores *deep learning*

En el Capítulo 2 hemos podido ver diferentes arquitecturas de *deep learning*. Todas ellas inspiradas en redes neuronales artificiales. Estas redes tienen dos etapas, una de extracción de características y otra de detección. Normalmente, una capa totalmente conectada al final de la red se encarga de la detección. El método que se propone en este trabajo para adaptar el sistema de detección de personas consiste en extraer los vectores de características directamente desde alguna capa anterior a la capa totalmente conectada. Para hacer esto se asumirá, como en otras investigaciones, que las características que ha aprendido la red para clasificar las imágenes serán útiles para nuestra tarea.

El método consiste en extraer las características directamente desde una CNN pre-entrenada. Existen modelos pre-entrenados con imágenes de diferentes bases de datos con un objetivo de clasificación y con diferentes arquitecturas.

3.2.1. Cambios en el esquema

Para extraer descriptores existen muchos modelos pre-entrenados de CNNs. Aquí vamos a ver qué redes han sido usadas en este trabajo para llevar a cabo el objetivo propuesto y cuáles son sus principales características. Después se explicará el criterio de selección de capas para extraer los descriptores y finalmente los cambios realizados sobre el esquema del sistema de detección de personas para llevar esto a cabo.

CNNs pre-entrenadas para extracción de características

- **AlexNet**

  Creada en 2012, [51], AlexNet tiene una arquitectura bastante similar a la desarrollada por Yann LeCun *et al.* para su red LeNet en [52], pero mucho más profunda, con capas convolucionales apiladas, y con más filtros en cada capa. Los principales logros de este trabajo son: el uso de ReLUs como función de activación en lugar de tangentes hiperbólicas para la adición de no-linealidad a los datos, *dropout* como método de regularización para evitar el *overfitting* y la superposición de capas de pooling para reducir el tamaño de la red.

  AlexNet tiene un total de 5 capas convolucionales y 3 capas totalmente conectadas (Figura 3.4). La ReLU se aplica después de todas las capas convolucionales y la regularización por dropout se aplica justo antes de la primera y segunda capa totalmente conectada. Por otro lado, los kernels o filtros de la red se encargan de extraer características de la imagen.

  AlexNet fue entrenada con la base de datos del proyecto ImageNet [53], diseñada especialmente para su uso en sistemas de reconocimiento de objetos y que contiene más de un millón de imágenes y 20,000 clases.

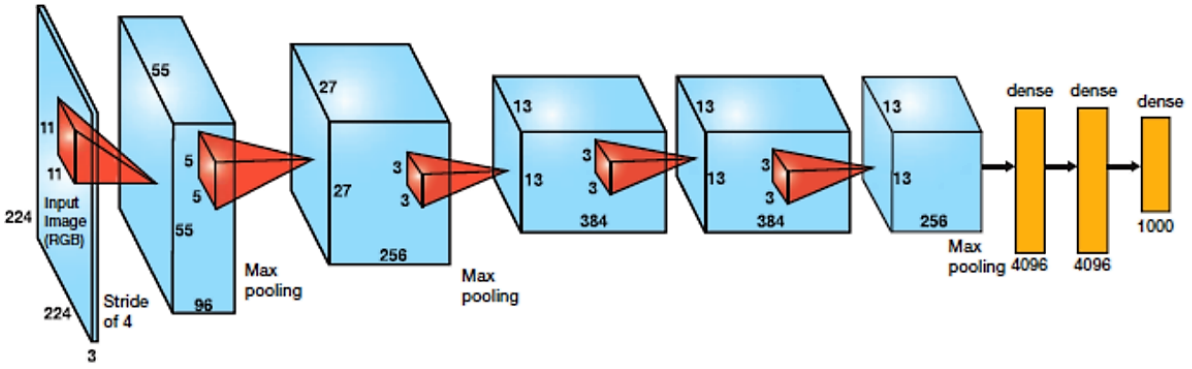

Figura 3.4: Arquitectura de la red AlexNet. Extraído de [9].

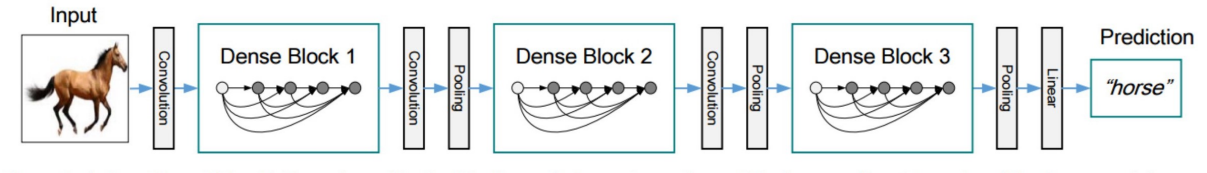

Figura 3.5: Arquitectura de la red DenseNet con tres bloques densos. Las capas de transición son las capas situadas entre bloques, donde se cambia el tamaño del mapa de características con convoluciones y *pooling*. Extraído de [10].

- **Densenet201**

  Al igual que AlexNet, la CNN pre-entrenada Densenet201 o *Densely Connected Convolutional Network* [54] también fue entrenada con la misma base de datos, ImageNet. Es capaz de clasificar más de 1000 clases de objetos y tiene una profundidad de 201 capas. Es uno de los modelos más profundos basados en CNNs.

  El largo recorrido que realiza la información a traves de una CNN muy profunda puede llegar a provocar la pérdida de esta antes de llegar al final. Para solucionar el problema, DenseNet simplifica las conexiones entre capas usadas en otras arquitecturas como ResNet [55], conectando cada capa directamente entre sí (Figura 3.5). De esta manera el número de parámetros que necesita la red en comparación con una CNN convencional se reduce, a pesar de ser más profunda. Esto lo hace reutilizando los mapas de características.

  En una red CNN normal, las capas se conectan en serie, una detrás de otra después de realizar un conjunto de operaciones (convoluciones, max pooling, etc.).

  La única condición que se debe cumplir es que los mapas de características deben tener el mismo tamaño. Para ello, la red se divide en bloques densos, donde el tamaño del mapa es constante. Las capas entre bloques se llaman capas de transición (Figura 3.5) y su número de filtros varía.

  Estas características hacen que este tipo de redes sean rápidas y fáciles de optimizar. Reutilizan características y en consecuencia se consiguen mejores aprendizajes y resultados mejores. Es una buena opción para la extracción de características, tal como apunta el trabajo que las dio a conocer [55].

- **Mobilenetv2**

  Mobilenet V2 [11] es una CNN cuya arquitectura está pensada para correr de manera muy eficiente en dispositivos móviles. Es una mejora de la red MobileNet V1 [56], que era muy básica y se utilizada para clasificar imágenes y extraer características.

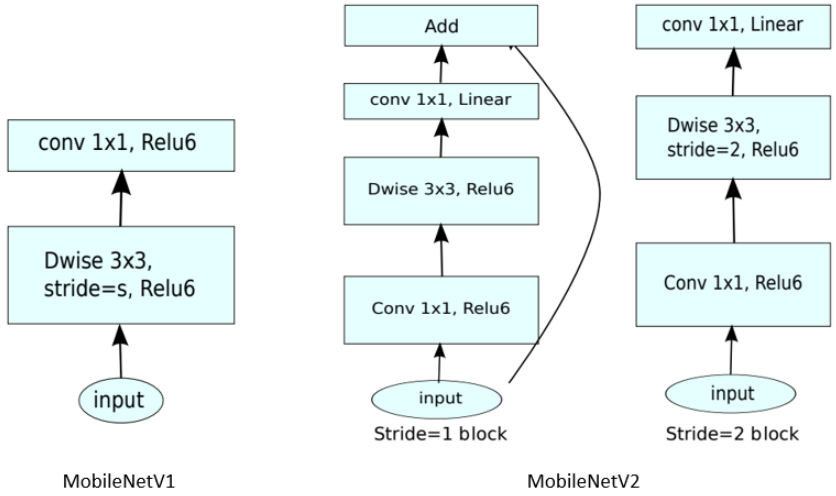

Figura 3.6: Bloques convolucionales de: MobileNetV1 a la izquierda y MobiletV2 a la derecha. Extraído de [11].

Para entender su arquitectura primero hay que hacer una revisión rápida a su primera versión. En ella, las capas convolucionales pueden ser sustituidas por las llamadas capas de convoluciones separables en profundidad (*depthwise separable convolutions*). Formadas por una convolución profunda, que filtra la entrada y después se aplica una convolución con un kernel de tamaño 1x1 (convolución puntual). Además, en las *depthwise separable convolutions* no hay capas de pooling Figura 3.6.

La versión V2 de la red ahora tiene un bloque compuesto por tres convoluciones en serie Figura 3.6, donde la convolución puntual al final del bloque hace justo lo contrario, reduce el número de canales. La primera capa del bloque será ahora una capa de expansión. El nombre con el que se la conoce es capa de cuello de botella (*bottleneck layer*), ya que hace más pequeñas las imágenes que recorren la red.

Estas características hacen que MobileNetV2 sea muy eficiente y también interesante para conocer su rendimiento con el sistema de adaptación propuesto.

- **VGG19**

  La red VGG fue introducida por Simonyan y Zisserman en 2014 [57]. Basada como todas la demás, en una CNN, destaca por ser una red bastante simple pero profunda. En el artículo de VGG se investiga el efecto de la profundidad en la red sobre los resultados para el reconocimiento de imágenes. Donde se concluye que esta profundidad juega un papel importante, las redes más profundas dan mejores resultados. Las imágenes de entrada a la red tienen un tamaño de 224x224x3. La arquitectura consiste en unas capas convoluciones apiladas aumentando la profundidad, gracias a un tamaño de kernel bastante pequeño, 3x3. Reduciendo el tamaño de los datos con capas de max pooling de 2x2 y dos capas totalmente conectadas al final de la red, seguidas de un clasificador softmax, Figura 3.7.

  Ofrece buenos resultados, pero es una red muy lenta de entrenar ya que contiene alrededor de 160 millones de parámetros, la mayoría de ellos situados en las capas totalmente conectadas del final.

3.2.2. Adaptación del sistema

El método propuesto, consiste en extraer los vectores de características a través de alguna de las capas de los modelos de CNN pre-entrenados que hemos visto en la sección 3.2.1. En

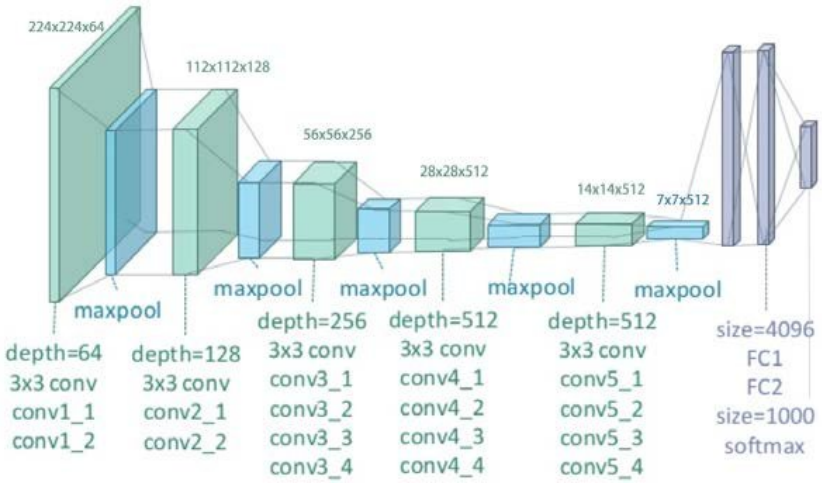

Figura 3.7: Estructura de VGG19. Extraído de [**?**].

este trabajo se ha usado el *Deep Learning Toolbox* de MATLAB para cargar las redes y sacar los vectores de características. Por lo general, se intentarán extraer los descriptores a partir de las salidas de las capas de max pooling, ya que estas se encargan de reducir el tamaño de las imágenes y son las últimas capas después de las capas convolucionales y las capas de función de activación, lo cual ayuda a intentar coger aquellas características más generalistas y robustas.

La extracción de los descriptores se hace de manera *offline*, es decir, primero se generan los descriptores para toda la base de datos en la herramienta de aprendizaje profundo de MATLAB, se guardan en memoria y después, los vectores son cargados en el código C++ con librerías de OpenCV que implementa el sistema. En la Figura 3.1 se puede ver el bloque adaptado en el sistema con un recuadro rojo. Este bloque se encargará de pasar los descriptores en la detección a la malla de SVM (Sección 3.1.2) y, por otro lado, en el entrenamiento los pasará a la etapa de asignación de etiquetas positivas y negativas de cada SVM.

Los descriptores se obtienen con las activaciones de la capa de la red convolucional. En primer lugar, las imágenes de la base de datos deben ser redimensionadas al tamaño de entrada de cada red, ya que estas tienen una resolución de 600x800 píxeles en la base de datos utilizada (PIROPO [8]). Las salidas de las capas suelen ser matrices de varias dimensiones, aunque, ya sean matrices multidimensionales o vectores, el método utilizado para la extracción de los descriptores es el mismo: concatenar cada fila de las matrices que se obtienen a la salida de la capa. De esta manera se consigue un descriptor para cada imagen, cuyo tamaño vendrá definido por el tamaño de los datos de cada capa. Esto hace posible adaptar el esquema para que, a partir de ahora, los tres métodos de extracción de características estén presentes en el mismo sistema, pudiendo cambiar entre uno y otro según las necesidades.

4

Experimentos y resultados

4.1. Introducción

En este capítulo se van a exponer los resultados obtenidos con los descriptores extraídos de cada una de las CNN pre-entrenadas de la Sección 3.2.1. También se detallarán las características de la base de datos empleada, así como las de las secuencias de entrenamiento y test. Se hará una comparación con los mejores resultados que se obtuvieron en [6] para el sistema PD. El tamaño de los descriptores y el coste computacional asociado al entrenamiento también se analizarán. Finalmente se mostrarán los resultados globales de rendimiento del sistema.

Las redes usadas, su número total de capas, las capas seleccionadas, su posición dentro de la red y el tamaño del descriptor obtenido se muestran en el Cuadro 4.1. Dada la gran cantidad de CNN presentes en el estado del arte, el criterio de elección de redes se ha hecho en base a las características de cada red. Algunas son más profundas como Densenet201 y otras más sencillas como AlexNet o MobileNetV2. Por otro lado, se han cogido tres capas para cada red, a diferentes profundidades con el fin de analizar qué influencia tiene cada una en los resultados. Para la red VGG19 el tamaño de los descriptores para capas menos profundas era demasiado grande, por lo que en este caso solo se obtendrán resultados de una capa.

Red	Nº total de capas	Capa	Nº de capa	Tamaño del descriptor
AlexNet	25	pool1	5	69998
		pool2	9	43264
		pool5	16	9216
Densenet201	709	pool3_pool	140	50176
		pool4_pool	480	43904
		avg_pool	706	1920
MobileNetV2	155	block_5_add	53	25088
		block_14_add	131	7840
		global_average_pool	152	1280
VGG19	47	pool5	38	25088

Cuadro 4.1: Redes, profundidad, capas utilizadas para la extracción de características, número de capa dentro de la red y tamaño del descriptor obtenido.

Nombre de la secuencia	Descripción
training	1 persona caminando exhaustivamente alrededor de la habitación
test1	1 persona (presente en la secuencia de entrenamiento) caminando
test2	Hasta 3 personas caminando a la vez
test3	1 persona (nueva para el sistema) caminando
test4	1 persona caminando y parada de pie
test9	1 persona caminando con condiciones de cambios de iluminación

Cuadro 4.2: Descripción de las secuencias de la base de datos PIROPO [8].

Todas las pruebas realizadas durante los experimentos se han hecho *off-line*, es decir, por un lado se han obtenido los descriptores *deep learning*, se han guardado como parte de la base de datos con cada una de sus etiquetas correspondientes y después se han cargado en el sistema para el entrenamiento y la detección. Con esto se quiere ver el rendimiento del sistema usando este tipo de descriptores. En un futuro se puede plantear incluir la extracción de características en el sistema.

4.2. Base de datos

La base de datos utilizada en el sistema de detección de personas ha sido PIROPO (*People in Indoor Rooms with Perspective and Omnidirectional cameras*) [8]. Se divice en dos habitaciones: La habitación *ROOM A* tiene un tamaño de 15x10 metros aproximadamente y cuenta con secuencias de 8 cámaras diferentes. Del conjunto de cámaras, 5 son de perspectiva, con una resolución de 704x576 a aproximadamente 10 fps, y 3 son omnidireccionales, con una resolución de 800x600 a aproximadamente 10 fps.

La habitación *ROOM B* tiene un tamaño de 8x5 metros aproximadamente y secuencias de 2 cámaras diferentes. Una de perspectiva con una resolución de 704x576 a aproximadamente 10 fps y otra omnidireccional con una resolución de 800x600 a aproximadamente 10 fps.

Los vídeos de todas las cámaras se dividen en secuencias de entrenamiento y de test. El número de personas en la sala, la persona que sale en cada secuencia y sus acciones vienen definidas en el Cuadro 4.2. En la base de datos existen más secuencias con diferentes descripciones, pero solo se presentan las que se han utilizado en este trabajo. Por otro lado, la cámara que se

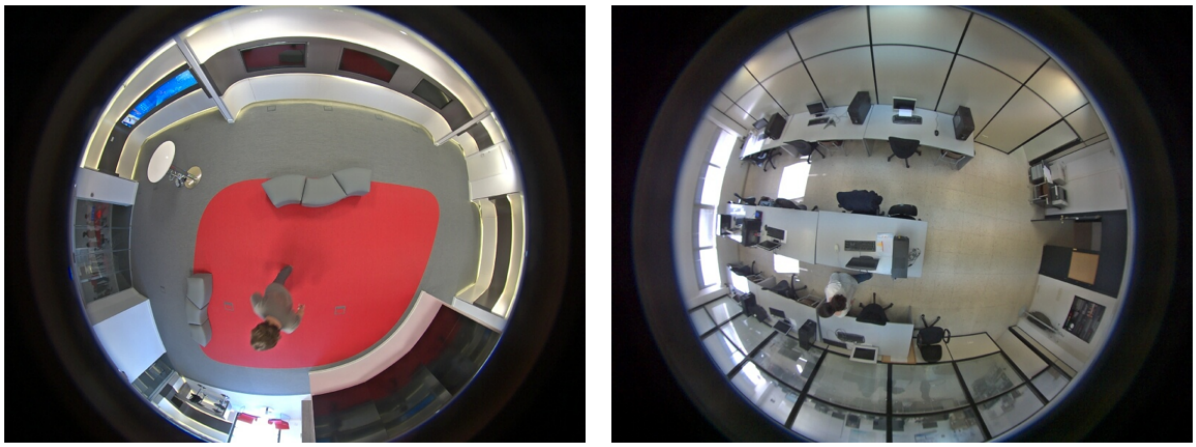

Figura 4.1: Imagenes de la base de datos PIROPO: A la izquierda habitación ROOM A y cámara OMNI 2A, a la derecha habitación ROOM B y cámara OMNI 1B. Extraído de [8].

ha usado para los experimentos ha sido la cámara omnidireccional *OMNI 2A* de la habitación *ROOM A*, una imagen de esta cámara se muestra en la Figura 4.1. Las grabaciones de esta habitación tienen algunas dificultades añadidas que pueden provocar problemas al sistema, como los reflejos en algunas paredes de cristal en la sala.

4.3. Evaluación

Para medir el rendimiento del sistema se utilizan diferentes medidas y parámetros, definidos a continuación.

Una tabla de clasificación binaria define cuatro medidas:

- **Verdadero positivo (TP)**: La clase real y la detectada son ambas positivas
- **Falso positivo (FP)**: La clase real es negativa pero la clase detectada es positiva.
- **Verdadero negativo (TN)**: La clase real y la detectada son ambas negativas.
- **Falso negativo (FN)**: La clase real es positiva pero la clase detectada es negativa.

En base a estas medidas se definen tres cálculos. La Precisión o ***Precision*** se define como la capacidad del clasificador de etiquetar como negativa una muestra que no es positiva, 4.1. El ***Recall*** o Recuperación es la capacidad para detectar todas las muestras positivas, 4.2. Estas dos medidas se utilizan para calcular el ***F-Score***, que se puede interpretar como una medida de rendimiento global del sistema (4.3).

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

Por otro lado, existen parámetros de diseño del sistema que pueden ajustarse para obtener mejores resultados. Estos son explicados a continuación.

En los esquemas basados en ventanas deslizantes la distancia entre el punto detectado y la etiqueta definida en el *ground truth* no está claramente definida, ya que en lugar de tener un píxel de referencia, se tiene una ventana que delimita una región de la imagen. En nuestro caso, se puede tomar como correcta una detección si la distancia entre el punto real, definido en el *gound truth*, y el punto detectado es mayor a un umbral, a partir de ahora definido como ***d***.

El **parámetro de regularización *C***, como dijimos en la Sección 2.2.3, es un parámetro de diseño que ajusta el hiperplano del SVM a los datos. Por tanto controla el *overfitting*: ajuste demasiado preciso del hiperplano a los datos de entrenamiento, empeorando los resultados con las secuencias de test.

4.4. Experimentos

4.4.1. Parámetros fijos

En el trabajo [6] donde se propuso el sistema de detección de personas se obtuvieron varias conclusiones sobre los resultados obtenidos. Se observó que el tipo de patrón de la malla de SVM

CAM	Sec	*C*	*Precision %*	*Recall %*	*F-Score %*
OMNI 2A	test1	**0.01**	**94,71**	**98,20**	**96,42**
		0.03	-0,78	+0,00	-0,41
		0.1	-0,59	+0,00	-0,30
		0.3	-0,59	+0,00	-0,30
	test2	**0.01**	**87,51**	**77,07**	**82,00**
		0.03	-0,66	-1,83	-1,37
		0.1	-0,81	-1,83	-1,43
		0.3	-0,81	-1,83	-1,43
	test3	**0.01**	**28,65**	**95,52**	**44,07**
		0.03	-4,00	+0,00	-4,87
		0.1	-4,77	-0,08	-5,87
		0.3	-4,77	-0,08	-5,87
	test4	**0.01**	**32,68**	**93,76**	**48,47**
		0.03	-5,38	-0,91	-6,27
		0.1	-5,61	-0,61	-6,52
		0.3	-5,61	-0,61	-6,52

Cuadro 4.3: Resultados con diferentes parámetros de regularización para la capa pool5 de AlexNet, *umbralMin*=2 y *umbralHiperplano*=0.5.

con el que mejores resultados se obtenían, para las imágenes omnidireccionales de la base de datos, era el patrón hexagonal, con un número de 825 clasificadores y tamaño de vecindario $Np = 7$. Por otro lado, en el trabajo se usó un valor de d=30, para poder comparar los resultados se escogerá este mismo valor en los experimentos. Además, Los mejores resultados se obtuvieron con el sistema PD, así que durante los experimentos, los resultados obtenidos en este trabajo se compararán siempre con ese sistema.

Estos parámetros se asumirán en un principio óptimos y serán usados por defecto en todos los experimentos realizados. Los parámetros que no se han fijado se evaluarán en la Sección 4.4.2 para encontrar el valor óptimo de cada uno de ellos. Para explicar el proceso se tomará como referencia la capa pool5 de AlexNet. Todos los experimentos hechos en esta capa y arquitectura serán análogos a todas las demás y los resultados se expondrán al final.

4.4.2. Resultados experimentales

Parámetro de regularización *C*

Vemos en el Cuadro 4.3 que los resultados obtenidos para C=0.01 en la capa pool5 de Alexnet son mejores en todos los casos, al igual que ocurría en el sistema PD de [6]. Para realizar este experimento se tienen que entrenar diferentes modelos para distintos valores de C. Dado el coste computacional de hacer esto para todas las demás capas y redes, el valor usado para el resto de experimentos será el mejor valor obtenido en la capa pool5 de AlexNet, C=0.01.

Umbrales

Como vimos en la Sección 3.1.2, cuando un vecindario de clasificadores realiza una detección devuelve un valor de confianza global para ese grupo. Podemos definir un umbral tal que si ese valor global no lo supera, la muestra se descarta. A partir de ahora nos referiremos a él como ***umbralMin***. Por otro lado, la función de pérdidas del SVM se puede modificar. Si el valor de confianza devuelto por el clasificador está por encima del cero, la muestra se considera positiva y si está por debajo negativa. El umbral de decisión en ese caso es el cero, pero podemos aumentarlo para evitar falsos positivos (valores positivos cercanos al cero). Nos referiremos a este

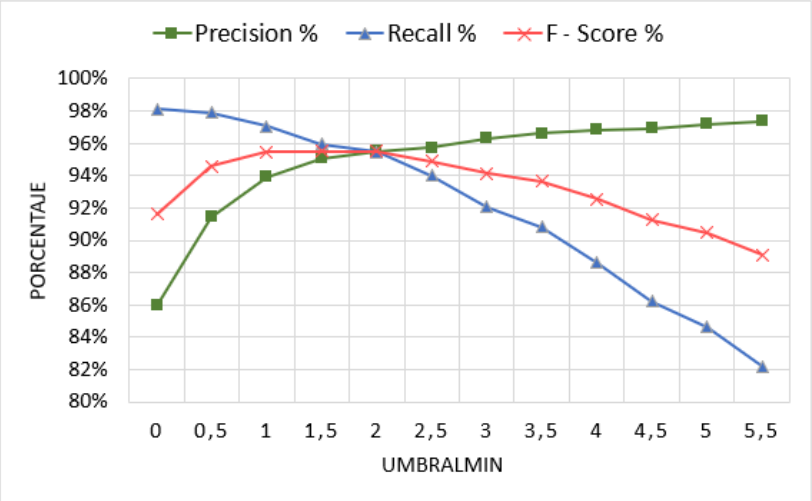

Figura 4.2: Resultados obtenidos para la secuencia test1, la capa pool5 de AlexNet, diferentes valores de *umbralMin* y *umbralHiperplano* = 0.

CAM	Sec	*umbralMin*	*umbralHiperplano*	*F-Score %*
OMNI 2A	test1	1.9	0	95,68
		1.9	0.2	96,21
		1.9	**0.4**	**96,59**
		1.9	0.6	96,03
		1.9	0.8	95,3
		1.9	1	94
		1.9	1,2	84,21

Cuadro 4.4: F-Score para la capa pool5 de AlexNet con los valores óptimos de *umbralMin* y *umbralHiperplano* del sistema PD.

umbral como ***umbralHiperplano***, debido a que intuitivamente, modificar este valor es análogo a desplazar el hiperplano entre las clases.

Al hacer experimentos con diferentes valores de *umbralMin* y *umbralHiperplano* la *Precision*, el *Recall*, y en consecuencia el *F-Score*, cambian. Generalmente, si la *Precision* aumenta, el *Recall* disminuye. La curva que define este relación depende del entrenamiento, de la base de datos y de la capa con la que se han extraído los descriptores. En la gráfica de la Figura 4.2 se puede ver como va aumentando la *Precision* para diferentes valores de *umbralMin*, mientras el *Recall* disminuye y el *F-Score* se mantiene entre las dos.

En cuanto al valor de los umbrales, en el trabajo [6] se observa que para el caso en el que *umbralMin* y *umbralHiperplano* son diferentes de cero, combinándolos adecuadamente, se obtienen mejores resultados. Concluyendo un valor óptimo *umbralMin*=1.9 y *umbralHiperplano*=1. Sin embargo, los experimentos que se han hecho en este trabajo con los descriptores extraídos de las CNN no cumplen del todo esta condición. Sobre la capa *pool5* de AlexNet se realizaron varios experimentos con los valores óptimos de los umbrales del sistema PD y se observó que los resultados sí mejoran al aumentar el *umbralMin*, pero suelen empeorar al aumentar mucho el *umbralHiperplano*, como se puede ver en el Cuadro 4.4. El procedimiento para obtener los umbrales óptimos tendrá en cuenta las secuencias *test1*, *test2*, *test3* y *test4*.

En el ajuste de los umbrales de cada red-capa se tomará como valor inicial *umbralHiperplano*=0, debido a las conclusiones anteriores. En este caso la optimización será independiente de cada parámetro, pero lo ideal sería hacerlo de manera conjunta. Para explicar el procedimiento se tomará de referencia la capa pool5 de AlexNet:

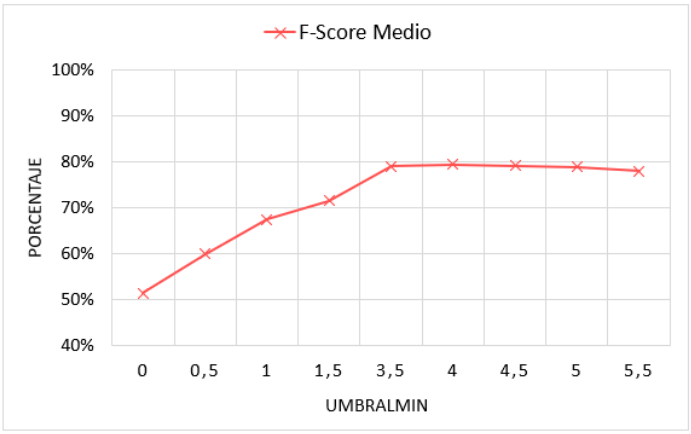

Figura 4.3: F-Score medio para la capa pool5 de AlexNet y *umbralHiperplano*=0.

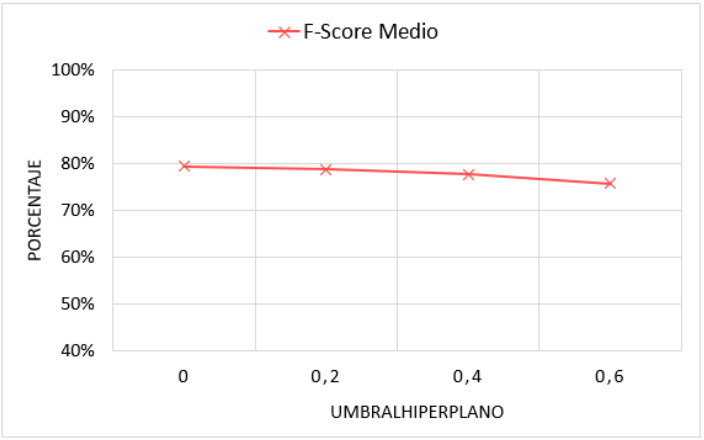

Figura 4.4: F-Score medio para la capa pool5 de AlexNet y *umbralMin*=4.

1. Variando el *umbralMin* para cada secuencia de test y fijando *umbralHiperplano*=0. Se hará una media de todos los *F-Score* obtenidos para cada *umbralMin* y se seleccionará el que mayor *F-Score* medio haya conseguido. En este caso se obtiene como óptimo *umbralMin*=4, Figura 4.3.

2. Se hará lo mismo pero para el *umbralHiperplano*. Se fijará el *umbralMin* óptimo obtenido en el paso anterior y se irá probando con diferentes *umbralHiperplano*, de nuevo para cada secuencia de test. Se hará la media de los resultados obtenidos y los umbrales óptimos serán aquellos que mejor media de *F-Score* hayan conseguido. En este caso se obtiene como óptimo *umbralHiperplano*=0, Figura 4.4.

3. Finalmente, el *F-Score* medio para la capa, será el resultado de promediar los *F-Score* obtenidos con los umbrales óptimos para cada secuencia. El Cuadro 4.5 representa este cálculo, donde se obtiene un F-Score=79.45 % para la capa pool5 de Alexnet.

El resto de medidas para las demás capas han sido obtenidas siguiendo el mismo procedimiento y serán expuestas en la Sección 4.4.4. Además, En el Cuadro 4.6 se representan todos los umbrales óptimos de cada red-capa. Se puede ver como para todas las capas el *umbralHiperplano* es más bajo que para el *umbralHiperplano* del sistema PD.

CAM	Red	Capa	Sec	*Precision %*	*Recall %*	*F-Score %*
OMNI 2A	AlexNet	pool5	test1	96,84	88,6	92,53
			test2	91,91	62,33	74,29
			test3	78,53	87,97	82,98
			test4	57,3	83,56	67,99
			Media	81,15	80,62	79,45

Cuadro 4.5: F-Score medio final de la capa pool5 AlexNet con *umbralMin*=4 y *umbralHiperplano*=0.

Red	Capa	*umbralMin óptimo*	*umbralHiperplano óptimo*
AlexNet	pool1	2	0
	pool2	2.5	0
	pool5	4	0
Densenet201	pool3_pool	1.5	0
	pool4_pool	2	0
	avg_pool	3	0.2
MobileNetV2	block_5_add	2	0
	block_14_add	3	0.2
	global_average_pool	2.5	0.2
VGG19	pool5	2.5	0.4

Cuadro 4.6: *umbralMin* y *umbralHiperplano* óptimos de cada capa.

Normalización

Cuando un SVM tiene a la entrada datos normalizados, su capacidad de entrenamiento mejora. Estos datos de entrada se normalizan con la distribución de los datos que se ha tenido en el entrenamiento, asumiendo que la distribución de los datos de las secuencias de test es igual que las de entrenamiento. En esta sección hemos comprobado si normalizando los datos de test de la misma manera en que se normalizaban en el entrenamiento se obtenían mejores resultados o no. En el Cuadro 4.7 podemos ver, como se podía esperar, que los resultados del SVM mejoran normalizando los datos. Por ello, en todos los experimentos se aplicará la normalización a los descriptores obtenidos de las CNN. En el trabajo [6] se hace también este mismo proceso.

4.4.3. Influencia de la profundidad de una red

Como ya mencionamos en la Sección 3.2, en las primeras capas de una CNN se obtienen los rasgos más específicos de una imagen, mientras que en capas más profundas las características son capaces de definir aspectos más complejos. Para evaluar la influencia de la profundidad de las capas de una red en el rendimiento del sistema, utilizaremos la red densenet201 para este experimento. Esta red tiene una profundidad de 709 capas, Cuadro 4.1. De todas ellas, se han obtenido resultados para 3 capas diferentes: avg pool, pool4 pool y pool3 pool. La capa pool3 pool es la número 140, pool4 pool la número 16 y avg pool la número 706, Cuadro 4.1. En la

CAM	Sec	Normalización	*Precision %*	*Recall %*	*F-Score %*
OMNI 2A	test1	No	45.82	86.58	59.93
		Sí	**78.63**	**89.75**	**83.83**

Cuadro 4.7: Resultados para la normalización de los descriptores de la capa pool5 de Alexnet del test1.

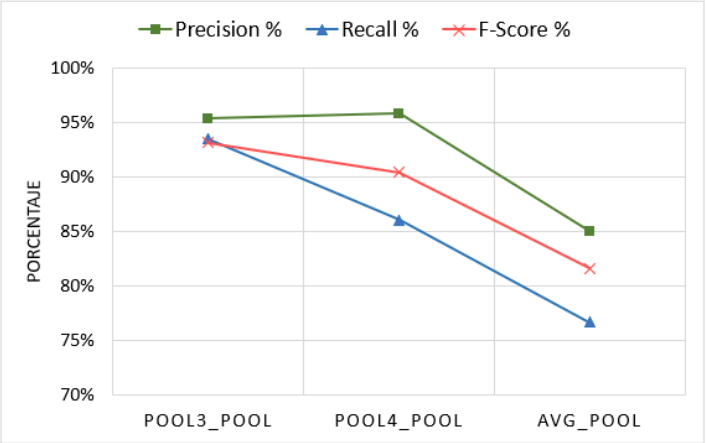

Figura 4.5: Precision, Recall y F-Score en función de la profundidad de la red densenet201.

Figura 4.5 se ve como varía la *Precision*, *Recall* y *F-Score* en función de la profundidad de la red, es decir, en función de las tres capas evaluadas.

Podemos concluir para este caso que se obtienen mejores resultados para las capas menos profundas de la red. La razón es difícil de definir, pero normalmente depende del tipo de aplicación para la que se esté usando la CNN. En este sistema para esta red se obtienen mejores resultados para la capa menos profunda, que tiene el tamaño de descriptores más grande. Lo ideal sería buscar un compromiso entre rendimiento y coste computacional, eligiendo como capa óptima la pool4 pool con un F-Socore=90.46 %.

4.4.4. Resultados globales

Finalmente, en el Cuadro 4.8 se muestran los resultados finales obtenidos para cada una de las capas. Todos estos resultados han sido calculados con los parámetros fijos y óptimos de cada capa, de la misma manera que se hizo en la Sección 4.4.2 con la capa pool5 de AlexNet. También se representan los mejores resultados del sistema PD para compararlos con los obtenidos en este trabajo.

El mejor resultado se obtiene para la capa pool3 pool de densenet201, con un F-Score=93.22 %. El sistema PD y descriptores HOG tiene de media un F-Score=96.2 %. La diferencia con este trabajo es de un 2.98 %. Para la capa de esta red se consiguen unos resultados muy próximos. Teniendo en cuenta que el tamaño del descriptor de la capa pool3 pool es 50176 y el del descripor HOG es de 86688, es interesante remarcar que el coste computacional que se obtiene en la etapa de detección con la red densenet201 sería menor que con los descriptores HOG. Además, los tiempos de entrenamiento también serían menores.

Por otro lado, cabe destacar los reultados obtenidos para la red MobileNetV2. Esta red está pensada para utilizarse en dispositivos móviles, por lo que su aquitectura y diseño es más sencillo. Como se podría esperar, en comparación con el resto de capas se obtienen los peores resultados. Sin embargo, teniendo en cuenta el objetivo de esta red, es interesante ver que se consigue un F-Score=87.30 % para la capa blobk 14 add.

4.4.5. Coste computacional del entrenamiento

El tiempo necesario para entrenar un clasificador suele ser bastante grande. En este caso no solo se entrena un SVM, sino una malla de SVM, concretamente 825 clasificadores. Este tiempo

CAM	Red	Capa	Sec	*Precision %*	*Recall %*	*F-Score %*
OMNI 2A	AlexNet	pool1	test1	99,77	93,5	96,53
			test2	99,21	83,05	90,41
			test3	40,96	86,07	55,5
			test4	61,71	91,78	73,8
			Media	75,51	88,6	80,81
		pool2	test1	97,96	93,65	95,75
			test2	96,77	80,37	87,81
			test3	43,33	92,45	59,01
			test4	82,56	92,23	87,13
			Media	80,16	89,68	80,86
		pool5	test1	96,84	88,6	92,53
			test2	91,91	62,33	74,29
			test3	78,53	87,97	82,98
			test4	57,3	83,56	67,99
			Media	81,15	80,62	79,45
	Densenet201	pool3_pool	test1	99,69	95,59	97,6
			test2	99,31	84,41	91,26
			test3	84,4	98,25	90,8
			test4	98,28	95,89	97,07
			Media	95,42	93,54	93,22
		pool4_pool	test1	99,46	93,93	96,62
			test2	98,53	73,37	84,11
			test3	87,8	93,69	90,65
			test4	97,67	83,25	89,89
			Media	95,87	86,06	90,46
		avg_pool	test1	98,54	92,56	95,46
			test2	92,49	49,82	64,76
			test3	81,29	88,3	84,65
			test4	67,84	76,1	71,73
			Media	85,04	76,70	81,62
	MobileNetV2	block_5_add	test1	99,52	91,34	95,25
			test2	99,57	81,77	89,8
			test3	58,96	91,04	71,57
			test4	99,66	91,93	95,64
			Media	89,43	89,02	85,54
		block_14_add	test1	99,02	94,87	96,90
			test2	98,07	73,1	83,76
			test3	70,67	95,52	81,23
			test4	85,61	91,47	88,44
			Media	88,34	88,74	87,30
		global_average_pool	test1	95,98	94,94	95,46
			test2	74,9	46,75	57,57
			test3	69,75	86,06	77,05
			test4	62,54	77,01	69,02
			Media	75,79	76,19	76,69
	VGG19	pool5	test1	94,85	95,81	95,33
			test2	86,77	73,45	79,56
			test3	40,24	91,54	55,90
			test4	35,87	91,62	51,56
			Media	64,43	88,11	76,93
	HOG (Sistema PD)		test1	98,6	97,9	98,2
			test2	98,7	86,6	92,3
			test3	97,8	96,5	97,1
			test4	98,2	96,1	97,1
			Media	98,3	94,2	96,2

Cuadro 4.8: Resultados globales.

Red	Capa	Tamaño del descriptor	Tiempo total de entrenamiento
AlexNet	pool1	69984	1 día, 1 hora
Densenet201	pool3_pool	50176	11.3 horas
MobileNetV2	block_5_add	25088	5.8 horas
	block_14_add	7840	1.5 horas

Cuadro 4.9: Coste computacional de la etapa de entrenamiento.

está directamente relacionado con el tamaño de los descriptores a su entrada. En el Cuadro 4.9 se da, para tres tamaños de descriptores diferentes, el tiempo que ha sido necesario emplear en entrenar la malla de clasificadores.

El tamaño de los descriptores obtenidos en este trabajo es pequeño en comparación con los descriptores HOG que se extraían en el sistema PD en [6]. Comparar los tiempos obtenidos con los del sistema PD no tiene mucho sentido, ya que los tiempos del sistema PD expuestos [6] dependen de parámetros como el número de clasificadores. Sin embargo, se quiere destacar que el tiempo más corto de entrenamiento en ese sistema fue de 1 día y 2 horas, mientras que en el sistema con descriptores extraídos de las CNN el menor tiempo es de 1.5 horas.

Las características del ordenador que se ha usado para hacer estas pruebas son: CPU ®Intel Core^{TM} i5 8600k @ 3.6 GHz, 16GB RAM con Windows 10 64bits.

5

Conclusiones y trabajos futuros

5.1. Conclusiones

Durante este trabajo se han realizado diferentes experimentos. De todos ellos se pueden sacar varias conclusiones.

En primer lugar, el principal problema del sistema que utiliza descriptores HOG es que para conseguir buenos resultados, se necesitan descriptores muy grandes. En este trabajo se ha demostrado que extrayendo descriptores de redes CNN pre-entrenadas, con capacidad de generalizar mejor y tamaños de descriptores más pequeños, se consiguen resultados igualmente buenos. Además, se debe tener en cuenta que el coste computacional es directamente proporcional al tamaño de los descriptores y, por esta razón, es interesante destacar que con los descriptores de aprendizaje profundo se consigue reducir también la carga computacional en las etapas de entrenamiento y test.

Observando los resultados globales, se puede ver que en general, se obtienen mejores resultados para las capas menos profundas de las redes. Esto quiere decir, que para este sistema y su aplicación, la detección de personas, funcionan mejor las capas mas tempranas de la red. Además, la profundidad de una red también parece tener impacto sobre los resultados globales. La red con la que mejores resultados se obtienen es densenet201, que precisamente es la red más profunda de todas las que se han analizado. También se puede ver que para capas menos profundas como AlexNet los resultados son aproximadamente un 10 % peores.

En cuanto al rendimiento por redes, destacan los resultados obtenidos para la red MobileNetV2, que a pesar de ser una red diseñada para ser usada en dispositivos móviles, se obtienen resultados que no distan mucho del caso mejor. En orden, la mejor red habría sido densenet201, seguida de MobileNet, en tercer lugar AlexNet y finalmente VGG19.

Los parámetros óptimos que se han calculado para cada capa han jugado un papel importante sobre los resultados. En el Cuadro 4.5 se puede ver que la variabilidad del *umbralMin* y del *umbralHiperplano* no es muy grande para diferentes capas. En general los resultados mejoran para valores en el rango de 2 y 3 para el *umbralMin* y rangos cercanos al cero para el *umbralHiperplano*. La variabilidad de estos umbrales para cada red, de nuevo no es grande, manteniéndose aproximadamente constante. Excepto para la red de AlexNet, que a pesar de que no varían mucho sus umbrales óptimos, sí lo hacen más que el resto de las redes, con lo cual hay que tener

en cuenta esto en el proceso de entrenamiento.

También podemos fijarnos en los resultados por secuencia. Como cabría esperar, con diferencia el mejor F-Score es para la secuencia de test1, ya que es la secuencia donde aparece la misma persona que en la secuencia de entrenamiento. Para la secuencia test2, donde aparecen varias personas a la vez también se obtienen en general buenos resultados, excepto para la capa global average pool de MobileNetV2 y avg pool de densenet, que son precisamente las dos capas más profundas de estas redes. Como hemos concluido antes, es de esperar que capas más profundas obtengan peores resultados. En últimos lugar, las secuencias test3 y test4 son las que mayor variabilidad presentan para las diferentes capas. Analizando cada una de las secuencias por separado, se podría concluir que, a pesar de que los resultados no son del todo malos, no llegan a alcanzar los obtenidos con descriptores HOG.

Concluimos que, a pesar de que en las pruebas iniciales, realizadas sobre la capa pool5 de Alexnet, no se viera que la CNN tuviera la suficiente capacidad de generalizar como para cambiar completamente el entorno y seguir sacando buenos resultados, los resultados globales obtenidos han sido similares a los que obtiene el sistema PD. Sin embargo, no los alcanza y todavía existe margen de mejora.

5.2. Trabajos futuros

Los trabajos futuros que se pueden realizar sobre este trabajo se basan en seguir haciendo experimentos. En la base de datos de PIROPO existen muchas más cámaras omnidireccionales con las que se puede valorar el rendimiento del sistema de detección de personas y así poder tener una visión más general de los resultados. Además, ya que existen secuencias de cámaras convencionales, también es interesante analizar los resultados que se obtendrían con este tipo de descriptores en estas cámaras. Esto implicaría además analizar el rendimiento del sistema en diferentes escenarios. Ya sea la forma de la sala, el movimientos de objetos en el fondo o cambios de iluminación. Finalmente, dado que todas las pruebas se han realizado extrayendo los descriptores de manera externa, podría implementarse esta extensión en el sistema, de forma que pudieran evaluarse los tiempos de ejecución en la detección y las posibilidades del sistema para trabajar en tiempo real.

Bibliografía

[1] Satya Mallick. Histogram of oriented gradients, 2016.

[2] David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258, 2009.

[3] Cathy Yeh. Support vector machines for classification. `http://efavdb.com/svm-classification/`, 2015.

[4] Akinori Hidaka and Takio Kurita. *Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks*, volume 2017. 2017.

[5] Arthur Ouakine. Review of deep learning algorithms for object detection. `https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852/`, 2018.

[6] L. García. P. Carballeira. Development of an algorithm for people detection using omnidirectional cameras. *Universidad Politécnica de Madrid, Proyecto de fin de carrera*, 2016.

[7] Xianghua Ying and Zhanyi Hu. *Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model*. 2004.

[8] C. R. del Blanco and P. Carballeira. The piropo database (people in indoor rooms with perspective and omnidirectional cameras). `https://sites.google.com/site/piropodatabase/`, 2016.

[9] Koustubh. Resnet, alexnet, vggnet, inception: Understanding various architectures of convolutional networks. `https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception//`, 2018.

[10] M. Chablani. Densenet. `https://towardsdatascience.com/densenet-2810936aeebb/`, 2017.

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[12] Research and Markets. Video surveillance market by system (analog, ip), offering (hardware, software, service), vertical (commercial, infrastructure, military defense, residential, public facility, industrial), and geography - global forecast to 2023. pages 1–194, 2018.

[13] Wei-Chih Hung Shengjin Wang Ming-Hsuan Yang Jingchun Cheng, Yi-Hsuan Tsai. Fast and accurate online video object segmentation via tracking parts. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7415–7424, 2018.

[14] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, PP:1–1, 11 2017.

[15] Khan Muhammad, Jamil Ahmad, Zhihan Lv, Paolo Bellavista, Po Yang, and Sung Baik. Efficient deep cnn-based fire detection and localization in video surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP, 03 2018.

[16] Eduardo Luz, Gladston Moreira, Luiz Antonio Zanlorensi Junior, and David Menotti. Deep periocular representation aiming video surveillance. *Pattern Recognition Letters*, 114:2–12, 2018.

[17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[18] Pedro F Felzenszwalb, David A McAllester, Deva Ramanan, et al. A discriminatively trained, multiscale, deformable part model. 2(6):7, 2008.

[19] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International journal of computer vision*, 38(1):15–33, 2000.

[20] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.

[21] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[22] Soojin Kim and Kyeongsoon Cho. Fast calculation of histogram of oriented gradient feature by removing redundancy in overlapping block. *J. Inf. Sci. Eng.*, 30(6):1719–1731, 2014.

[23] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. *Ten years of pedestrian detection, what have we learned.* 2014.

[24] Pramuditha Perera and Vishal M Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 2019.

[25] Di-Xiu Xue, Rong Zhang, Hui Feng, and Ya-Lei Wang. Cnn-svm for microvascular morphological type recognition with data augmentation. *Journal of medical and biological engineering*, 36(6):755–764, 2016.

[26] Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning strong parts for pedestrian detection. pages 1904–1912, 2015.

[27] Chunhui Gu. *Recognition using regions.* 2012.

[28] Francesco Comaschi, Sander Stuijk, Twan Basten, and Henk Corporaal. *RASW: A run-time adaptive sliding window to improve Viola-Jones object detection.* 2013.

[29] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. *An active search strategy for efficient object class detection.* 2015.

[30] Hui-Xing Jia and Yu-Jin Zhang. Fast human detection by boosting histograms of oriented gradients. pages 683–688, 2007.

[31] UC. Berkeley Ross Girshick. *Deformable part models.* 2013.

[32] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. *An HOG-LBP human detector with partial occlusion handling.* 2009.

[33] Guolong Gan and Jian Cheng. *Pedestrian detection based on HOG-LBP feature.* 2011.

[34] Jingsong Xu, Qiang Wu, Jian Zhang, and Zhenmin Tang. Fast and accurate human detection using a cascade of boosted ms-lbp features. *IEEE Signal Processing Letters*, 19(10):676–679, 2012.

[35] Wiebe Van Ranst, Floris De Smedt, and Toon Goedemé. *GPU Accelerated ACF Detector.* 2018.

[36] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. *Predicting time series with support vector machines.* 1997.

[37] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[38] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[39] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation.* 2014.

[41] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[42] Ross Girshick. *Fast r-cnn.* 2015.

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster r-cnn: Towards real-time object detection with region proposal networks.* 2015.

[44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You only look once: Unified, real-time object detection.* 2016.

[45] Peiming Ren, Wei Fang, and Soufiene Djahel. A novel yolo-based real-time people counting approach. In *2017 International Smart Cities Conference (ISC2)*, pages 1–2. IEEE, 2017.

[46] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, Joao Barreto, et al. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(1–2):1–183, 2011.

[47] Stephen T Barnard. Interpreting perspective images. *Artificial intelligence*, 21(4):435–462, 1983.

[48] Wolfgang Schulz, Markus Enzweiler, and Tobias Ehlgen. *Pedestrian recognition from a moving catadioptric camera.* 2007.

[49] Yoshio Onoe, Naokazu Yokoya, Kazumasa Yamazawa, and Haruo Takemura. *Visual surveillance and monitoring system using an omnidirectional video camera*, volume 1. 1998.

[50] Zoran Zivkovic et al. *Improved adaptive Gaussian mixture model for background subtraction.* 2004.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* 2012.

[52] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[53] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. *Imagenet: A large-scale hierarchical image database*. 2009.

[54] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. *Densely connected convolutional networks*. 2017.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition*. 2016.

[56] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.